DTU Electrical and Photonics Engineering
Department of Electrical and Photonics Engineering

Accelerated methods for computing acoustic sound fields in dynamic virtual environments with moving sources

Nikolas Borrel-Jensen
PhD Thesis
Kongens Lyngby, August 2023
(Revised November 2023)

Accelerated methods for computing acoustic sound fields in dynamic virtual environments with moving sources

Supervisor: Associate Professor Cheol-Ho Jeong
Co-supervisors: Associate Professor Allan P. Engsig-Karup & Professor Maarten Hornikx

Doctoral thesis in Electrical and Photonics Engineering, August 2023

By Nikolas Borrel-Jensen

Cover photo: AI-generated with DALL-E 2 using the prompt 'Digital art of a cow with a bell around its neck on a cliff wearing a VR headset,' 2023.
"As the sun sets, the cow dons a kaleidoscopic VR headset, entering a technicolor meadow where neon flowers sway to cosmic music. It rides a luminescent rainbow, flies through exploding stars, and dances with morphing animals. In an underwater disco, it grooves with light beings and swims alongside neon fish. The VR world bends time and space, showcasing historical and futuristic realms. Finally, the cow becomes part of a cosmic cow constellation, connecting to the universe's rhythmic energy. As it removes the headset, the world remains transformed by the surreal, colorful journey."
- written by ChatGPT, 2023.

Published by: DTU, Electrical and Photonics Engineering, Ørsteds Plads, Building 352, 2800 Kongens Lyngby, Denmark
`https://electro.dtu.dk`
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Preface

This PhD thesis was prepared at the Department of Electrical and Photonics Engineering at the Technical University of Denmark (DTU) in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD) in Electronics and Communication. The work presented in this thesis was completed between December 1st, 2019, and July 31st, 2023 at the Acoustic Technology group, department of Electronics and Photonics Engineering, DTU, Lyngby, under the supervision of Associate Professor Cheol-Ho Jeong, Associate Professor Allan P. Engsig-Karup and Professor Maarten Hornikx.

Acknowledgements

Before diving into my PhD studies, I remember having thoughts about why I should commit to three intense years of academic rigor. After much introspection, I narrowed it down to a few reasons: my desire to expand my knowledge in acoustics, to connect with inspiring and enjoyable individuals, my eagerness to experience an internationally esteemed university, my aspiration to intern at a major tech company, and last, my goal to earn my PhD degree. I feel fortunate to have achieved everything on that list except my degree, which I am hopefully well on my way to achieving.

My biggest thanks go to my main supervisor Cheol-Ho. You have always prioritized giving careful, detailed, and very useful feedback, especially when I have felt it difficult. I appreciate the freedom and support you have shown me. Further, I would like to thank my supervisor Allan. You have been a big inspiration and shown the way to many new and exciting research topics, which has helped me find the motivation. I learned a lot from you, also in the sweat-dripping courses in advanced numerical methods. I would also like to thank my last supervisor, Maarten. Although our long-distance relationship has been challenging at times, I highly appreciate your commitment down to the smallest detail. I also enjoyed your hospitality when visiting your group at TU/Eindhoven.

I want to say a special thanks to George, who accepted me as a visiting research scholar at the CRUNCH group at Brown University. I knew the academic level was high, but I did not know it was also such a welcoming environment. I could not have found a better place to visit. I would like to thank all in the CRUNCH group, especially Ahmad, Rômulo, Ehsan, Adar, Joseph, Kasia, and Somdatta.

Also, I would like to thank Sebastian from Meta for hiring me as an intern; it was a pleasure to work with you and meet so many extraordinary people in the lab. Thanks to Nils, Andrew, Alexander, Setare, Gregor, and everyone else. Thanks to Ben and Louise for being so kind and helpful, giving us a perfect start in Redmond.

Thanks to all my colleagues and friends at DTU! Especially my office mates Hermes, Sophia, and Finnur. The environment in the acoustics group has been very pleasant also due to Finn's kindness as head of the department.

I would like to express my gratitude to my parents. Without my mom's unwavering reassurance and my dad serving as a big inspiration, I would not have been able to achieve what I have.

Last but not least, thanks to Katharina for your support, unbelievable love, and patience - it was a memorable six months in Redmond!

Abstract

Realistic sound is essential in virtual environments, such as computer games, virtual and augmented reality, metaverses, and spatial computing. The wave equation describes wave phenomena such as diffraction and interference, and the solution can be obtained using accurate and efficient numerical methods. Due to the often demanding computation time, the solutions are calculated offline in a pre-processing step. However, pre-calculating acoustics in dynamic scenes with hundreds of source and receiver positions are challenging, requiring intractable memory storage.

This PhD thesis examines novel scientific machine learning methods to overcome some of the limitations of traditional numerical methods. Employing surrogate models to learn the parametrized solutions to the wave equation to obtain one-shot continuous wave propagations in interactive scenes offers an ideal framework to address the prevailing challenges in virtual acoustics applications. Training machine learning models often require a large amount of data that can be computationally expensive to obtain; hence this PhD thesis also investigates efficient numerical methods for generating accurate training data.

This study explores two machine learning methods and one domain decomposition method for accelerating data generation. (1) A physics-informed neural network (PINN) approach is taken, where knowledge of the underlying physics is included in the model, contrary to traditional 'black box' deep learning approaches. A PINN method in 1D is presented, which learns a compact and efficient surrogate model with parameterized moving source and impedance boundaries satisfying a system of coupled equations. The model shows relative mean errors below 2%/0.2dB and proposes a first step towards realistic 3D geometries. (2) Neural operators are generalizations of neural networks approximating *operators* instead of approximations of *functions* typical in deep learning. The DeepONet is a specific framework used in this thesis for operator learning applied to approximate the wave equation operators. The proposed model enables real-time prediction of sound propagation in realistic 3D acoustic scenes with moving sources, avoiding the offline calculation and storage of impulse responses. Our computational experiments, including various complex 3D scene geometries, show good agreement with reference solutions, with root mean squared errors ranging from 0.02 Pa to 0.10 Pa. Notably, our method signifies a paradigm shift as no prior machine learning approach has achieved precise predictions of complete wave fields within realistic domains. (3) A rectangular domain decomposition method is proposed, enabling error-free sound propagation in the bulk of the domain consisting of air. The Fourier method exploits the known analytical solution to the wave equation in the rectangular domain with near-optimal spatial discretization satisfying the Nyquist criterium via the Fast Fourier Transform for calculating derivatives. By coupling the Fourier method with the spectral element method (SEM) near the boundaries, the method is capable of handling complex geometries with the caveat of introducing errors at the coupling interface. A significant speed improvement is reported for a 1D domain when

the domain decomposition method is used instead of the SEM running in the full domain.

Keywords: virtual acoustics; physics-informed neural networks (PINNs); neural operators; DeepONet; transfer learning; domain decomposition; Fourier methods; spectral element method; high-performance computing; real-time computing.

Resumé

Realistisk lyd er afgørende i virtuelle miljøer som computerspil, virtuel og udvidet virkelighed (VR/AR), metaverser og 'spatial computing'. Den akustiske bølgeligning beskriver fænomener som diffraktion og interferens, og kan løses effektivt og nøjagtigt med numeriske metoder. Da løsningerne ofte er beregningstunge, beregnes disse i et præ-processeringsstep. Dog er præ-beregning af akustik i dynamiske scener med hundreder af kilde- og modtagerpositioner udfordrende, da det kræver stor lagerkapacitet.

Dette ph.d.-projekt undersøger nyskabende videnskabelige metoder til maskinlæring med det formål at overvinde nogle af begrænsningerne i traditionelle numeriske metoder. Ved at anvende surrogatmodeller til at lære parametriserede løsninger af bølgeligningen kan fremkaldelsen af bølgeudbredelsen i interaktive scener ske med en enkelt forespørgsel og er derfor en ideel teknik til håndtering af de nuværende udfordringer inden for anvendelser i virtuel akustik. Træning af maskinlæringsmodeller kræver ofte store mængder data, der kan være beregningsmæssigt dyrt at generere; derfor undersøger dette ph.d.-projekt også effektive numeriske metoder til at generere præcise træningsdata.

To maskinlæringsmetoder og en domæne-dekompositionsmetode til mere effektiv datagenerering er blevet udforsket i denne undersøgelse. (1) En metode kaldet physics-informed neural network (PINN) er anvendt, hvor viden om den underliggende fysik er inkluderet i modellen. Dette står i modsætning til traditionelle dybe neurale 'black box' netværk. En PINN-metode i 1D præsenteres, hvor en kompakt og effektiv surrogatmodel med parametriserede bevægelige lydkilder og impedans-randbetingelser lærer et system af koblede ligninger. Modellen har relative gennemsnitlige fejl under 2%/0.2dB og foreslås som et første skridt mod realistiske 3D-geometrier. (2) Neurale operatorer er generaliseringer af neurale netværk, der approksimerer *operatorer* i stedet for *funktioner* typisk for dybe neurale netværk. DeepONet er en specifik metode til operator-læring og er blevet anvendt i dette ph.d.-projekt til at approksimere bølgeligningsoperatorer. Den foreslåede model muliggør realtidsforudsigelser af lydudbredelsen i realistiske 3D-scener med bevægelige lydkilder og undgår dermed offline-beregninger samt lagring af de beregnede impulsresponser. Vores beregningsmæssige eksperimenter, inklusiv diverse komplekse 3D-geometrier, viser god overensstemmelse med referenceløsninger, med kvadratisk middelfejl (RMSE) på mellem 0.02 Pa og 0.10 Pa. Vores metode repræsenterer et paradigmeskift, da ingen tidligere maskinlæringsmetode har opnået præcise forudsigelser af fuldstændige bølgefelter i realistiske domæner. (3) En rektangulær domæne-dekompositionsmetode er blevet foreslået, der muliggør fejlfri lydudbredelse i størstedelen af domænet tilsvarende luft. Fourier-metoden udnytter en diskretisering af den kendte analytiske løsning på bølgeligningen i det rektangulære domæne med næsten-optimal rumlig diskretisering i overensstemmelse med Nyquist-kriteriet via den hurtige Fourier transformation til beregning af de afledte. Ved at koble Fourier-metoden med spektralelement-metoden (SEM) nær grænserne er metoden i stand til at håndtere komplekse geometrier, dog med det

forbehold, at fejl introduceres ved koblingsgrænsefladen. En betydelig hastighedsforbedring er rapporteret i et 1D-domæne, når domæne-dekompositionsmetoden er anvendt i stedet for SEM i hele domænet.

Nøgleord: virtuel akustik; fysik-informerede neural netværk; neurale operatorer; DeepONet; lærings-overførsel; domæne-dekomposition; Fourier-metoder; spektralelement-metode; højtydende databehandling; realtidsberegning.

List of publications

The present PhD thesis is based on a collection of scientific articles: Paper A to Paper D. Paper A is published in a scientific journal, Paper B is submitted to a scientific journal, and Paper C and D are conference articles.

- **Paper A**. Nikolas Borrel-Jensen, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Dec. 2021). "Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries." en. In: *Journal of the Acoustical Society of America, Express Letters* 1.12, page 122402.

- **Paper B**. Nikolas Borrel-Jensen, Somdatta Goswami, Allan P. Engsig-Karup, George Em Karniadakis, and Cheol-Ho Jeong (2023). "Sound propagation in realistic interactive 3D scenes with parameterized sources using deep neural operators." en. In: *Submitted*.

- **Paper C**. Nikolas Borrel-Jensen, Allan P. Engsig-Karup, and Cheol-Ho Jeong (2023). "A sensitivity analysis on the effect of hyperparameters in deep neural operators applied to sound propagation." en. In: *10th Convention of the European Acoustic Association of Forum Acusticum 2023*.

- **Paper D**. Nikolas Borrel-Jensen, Allan P. Engsig-Karup, Maarten Hornikx, and Cheol-Ho Jeong (Aug. 2022). "Accelerated sound propagation simulations using an error-free Fourier method coupled with the spectral element method." In: *Proceeding of Inter.noise 2022, Glasgow, Scotland*.

The following technical reports complement the thesis: Technical Report A has overlapping content with Paper D but gives more insights into interpolation techniques, the finite-difference time-domain scheme, and the spectral element method and provides more extensive results; Technical Report B summarizes the wave-based boundary element method including a short convergence study.

- **Report A**. Nikolas Borrel-Jensen (2023a). *Accelerated sound propagation simulations using an error-free Fourier method coupled with the spectral element method*. Technical report. Technical University of Denmark.

- **Report B**. Nikolas Borrel-Jensen (2023b). *A summary of wave-based boundary element method (WBBEM) with convergence studies*. Technical report. Technical University of Denmark.

This thesis does not include the following: Paper E is redundant, and Abstract A consists of an abstract of the conference talk.

- **Paper E**. Nikolas Borrel-Jensen, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Oct. 2022a). "DeepONet: Learning the sound propagation in 1D with parameterized sources using deep learning for approximating the wave equation operators." en. In: *Proceeding of the 24th International Congress on Acoustics*.

- **Abstract A**. Nikolas Borrel-Jensen, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Apr. 2022b). "Machine learning-based room acoustics using flow maps and physics-informed neural networks." In: *Presented at the 182nd Meeting of the Acoustical Society of America, 23-26 May, Denver* 151, A232–A233. ISSN: 0001-4966.

The code for reproducing all results related to this PhD can be found in Code A to Code D. Code A contains Python code reproducing results for Paper A; Code B contains Python code reproducing results for Paper B and Paper C (will be available upon publication); Code C contains Matlab code for reproducing results from Paper D and Report B, and was used to generate data for Paper C; and code D contains C++ code for solving the 3D wave-equation using a DG-FEM solver developed by (Melander et al. 2020).

- **Code A**. https://github.com/dtu-act/pinn-acoustic-wave-prop

- **Code B**. https://github.com/dtu-act/deeponet-acoustic-wave-prop

- **Code C**. https://github.com/dtu-act/numerical-pde-solvers

- **Code D**. https://github.com/dtu-act/libparanumal

Contents

CHAPTER 1

Introduction

1.1 Motivation

Reproducing the acoustics of our surroundings is pivotal for realistic and immersive experiences in virtual environments. Humans interact with their surroundings in a multisensory manner, where auditory cues collectively give humans a rich understanding of their surroundings. These cues (Plack 2018) involve the ability to localize sound sources using the differences in arrival time and intensity between ears together with the directivity pattern of the ears; estimate distances from intensities and spectral contents; gather information from the reverberation about the size, shape, and material properties of a space; and perceive the movement of sound sources by tracking changes in loudness, frequency content, and timing of sound.

Virtual acoustics is a widely used term referring to the simulation and replication of realistic acoustic environments modeled using algorithms and digital signal processing techniques. The aim is to recreate the auditory cues to give the listener the impression of being in a specific physical space. This has many applications in computer games, virtual reality (VR), augmented reality (AR), and architectural acoustics. Historically, auditory reproduction has been neglected compared to visual reproduction due to hardware limitations, where game developers had to prioritize resource usage. However, virtual acoustics is gaining more and more attention these days, driven especially by big tech companies. Meta has one of the largest audio research teams in the world (Meta 2020) investing heavily in the 'metaverse,' where one of the challenges is to adapt sound sources originating from other environments to the physical environment in which the sound has to be reproduced; Meta also introduced the Ray Ban Stories (Meta 2021) smart-glasses allowing users to capture a moment with the build-in camera or listen to music; Apple has probably been the largest driver in coining the term 'spatial audio' for the broader audience referring to the technique of creating and reproducing sounds in a 3D space through the support of the AirPod Pro earbuds (Apple 2022) for enriching films and music; and lately, spatial computing[1] (Greenwold 2003) has entered the scene with the Quest 3 and Quest Pro by Meta (Meta 2022) and the Vision Pro by Apple (Apple 2023).

With spatial computing, the trend is toward more immersive experiences where auditory perception will be crucial. There are currently two main approaches to interactive virtual acoustics: an entirely dynamic scenario with real-time acoustic simulations; or pre-calculated acoustics with conditions defined in a prior stage. For an entirely dynamic scenario, geometrical acoustics (GA) (Savioja et al. 2015) are typically applied due to their capabilities of executing below a real-time threshold of around 100 ms (Sandvad 1996). Several state-of-the-art methods exist focusing primarily on computer games (Wefers et al. 2018; Pelzer et al. 2014;

[1]The term 'spatial computing' was defined in 2003 by Simon Greenwold as 'human interaction with a machine in which the machine retains and manipulates referents to real objects and spaces.'

Lauterbach et al. 2007; Schröder 2011) with implementations in academic tools, e.g., *RAVEN* (Raven 2023) and commercial tools, e.g., *Oculus spatializer* (spatializer 2023), *Steam Audio* (Audio 2023), *Google Resonance* (Resonance 2023), *Audiokinetic Wwise* (Wwise 2023), and *FMOD* (Fmod 2023). Requiring real-time evaluation poses an accuracy limit for such methods. When fully dynamic scenes are not required, the so-called *wave-based methods* can be applied, taking the underlying physics into account in terms of the wave equation

$$\frac{\partial^2 p(\mathbf{x},t)}{\partial t^2} - c^2\nabla^2 p(\mathbf{x},t) = F(\mathbf{x},t), \qquad t \in \mathbb{R}^+, \qquad \mathbf{x} \in \mathbb{R}^N, \tag{1.1}$$

where p is the pressure (Pa), t is the time (s), c is the speed of sound in air (m/s), $F(\mathbf{x},t)$ is a forcing function and N is the dimensionality. Initial and boundary conditions should also be defined, determining the initial sound and the materials on the surfaces. Wave-based methods can be implemented by discretizing the domain in a grid of points where impulse responses for all source/receiver pairs are pre-calculated. Several state-of-the-art methods exist within acoustics such as finite element methods (FEM) (Okuzono et al. 2014), spectral element methods (SEM) (Pind et al. 2019), discontinuous Galerkin finite element methods (DG-FEM) (Melander et al. 2020) and finite-difference time-domain methods (FDTD) (Botteldoorena 1995; Hamilton et al. 2017). The number of commercial tools for the wave-based method is more limited compared to GA methods. Microsoft has been promoting 'Project Acoustics' (Microsoft 2023) as a plugin for Unity (Unity 2023) and Unreal (Unreal 2023) game engines for physically modeling how sound propagates within a scene taking its shape and materials into consideration and Treble Technologies (Treble 2023) recently introduced a new software tool for architectural design for accurately and efficiently simulating the acoustics allowing the user to experience the acoustics auditorily. One limitation of these methods is the necessity to store a large number of impulse responses, which can lead to significant storage requirements. As a result, there might be a trade-off that involves reducing the flexibility of the scene.

1.2 Scope for this work

This PhD study aims to develop efficient and accurate methods for reproducing the acoustic sound field in dynamic virtual environments with moving sources without the need for offline pre-calculating and storing all source/receiver impulse response pairs. The target applications are in the field of virtual and augmented reality, computer games, metaverses, and spatial computing. The study is particularly concerned about sound fields in closed cavities, such as rooms and buildings, where the boundary geometry and material properties are considered, focusing on accurately modeling the lower frequency sound field. Two main ideas are explored:

- We investigate if scientific machine learning models can accurately and efficiently predict the acoustic sound field in dynamic scenes with moving sources. By learning a compact and efficient surrogate model, impulse responses for any source/receiver pair should be predicted in real-time to be useful for the virtual acoustics applications of

interest. This is a paradigm shift compared to traditional numerical methods, where impulse responses for all source/receiver pairs are calculated offline and stored in a lookup table. The study corresponds to Paper A, Paper B, and Paper C.

- We investigate if domain decomposition methods can improve the efficiency of numerically solving the wave equation. Solving the wave equation is computationally expensive, even when using state-of-the-art numerical methods. In room acoustics, the predominant medium is air. By decomposing the domain into rectangular partitions consisting of air and geometrically flexible partitions near the boundaries, specialized methods can be employed in each partition. The study corresponds to Paper D.

1.3 Thesis structure

The present PhD thesis is based on a collection of scientific articles: Paper A to Paper D. Paper A is published in a scientific journal ('Editor's Pick'); Paper B is submitted to a high-impact journal; and Paper C and Paper D are conference articles. Additionally, Report A complements Paper D with additional information, and Report B includes background material for discontinued research. The rest of the thesis is organized as follows: Chapter 2 discusses the requirements for the applications of interest, outlines the assumptions, and gives an overview of current state-of-the-art methods; Chapter 3 discusses important features numerical methods should possess and explains the reasoning for researching domain decomposition methods and summarizes the interface handling; Chapter 4 introduces scientific machine learning and discusses how to overcome the challenges of handling dynamic scenes; and Chapter 5 presents conclusions drawn from the PhD study and discusses potential next steps of research.

CHAPTER 2

Requirements, assumptions, and methods for interactive applications

2.1 Requirements and assumptions

We have divided the requirements for games, VR/AR, and spatial computing applications into four groups: Usability, Interactivity, Authenticity, and Resource Usage. The usability requirements are related to the preparation of the environment and the user handling this (typically a sound designer); the interactivity requirements relate to how much the environment can change at run-time; the interactivity requirements are related to the acoustical details the user needs for the given application; and the resource usage requirements are related to the specific hardware and how much resources are being allocated to the auditory parts. Each of the requirements is considered below:

Usability. In current game and AR/VR development workflows, sound designers are often burdened with manually setting trigger boxes and designing filters for the space's acoustics (Turner et al. 2022). Our developed methods aim to automate some of these cumbersome tasks, freeing up more time for sound designers to focus on artistic aspects. By preparing scenes with boundary materials and geometries, our automated methods should enable sound designers to retrieve real-time impulse responses that accurately reflect the room conditions.

Interactivity. The scenes in these applications can be highly dynamic, with numerous moving sound sources and receivers. Moreover, the geometry may change at runtime due to actions like opening/closing doors, moving/removing walls, and altering obstacles.

Authenticity. While exact physical reproduction is not necessary, the acoustical properties of the environment should provide plausible and realistic soundscapes. Smooth sound field transitions are crucial when sound sources and receivers move within the scene.

Resource Usage. Games have strict computation and storage efficiency requirements, with most resources allocated to the graphics pipeline, leaving limited resources for audio processing. The typical budget for storing auditory assets, including impulse responses, is in the hundreds of MB, and audio calculations must be kept under a certain time threshold per frame.

Continuing from the introduction, the simulation methods for acoustic sound fields can be divided into geometrical acoustics (GA) and wave-based methods. GA approximates sound propagation as rays, neglecting wave phenomena like diffraction and interference. It assumes that sound rays travel in straight lines until they interact with obstacles, where they may be reflected, absorbed, or transmitted. This approximation is valid when the sound wavelength is much smaller than the obstacles, i.e., well above the Schroeder frequency. GA methods are more computationally efficient than wave-based methods but lack accuracy in certain scenarios.

On the other hand, wave-based methods account for the full wave nature of sound, considering diffraction, interference, and scattering by numerically solving the underlying wave equation. However, these methods demand extensive computation time and require the domain to be discretized well above the Nyquist limit to minimize discretization errors.

The choice between GA and wave-based methods depends on the available computational budget and the required accuracy. For applications where realistic simulations of the lower frequency sound field are essential, wave-based methods are preferred, as they can capture the full sound field accurately. Along with the requirements for the applications of interest, we have made the following assumptions:

Assumption 1 To accommodate the computational demands, we allow for offline calculations, spreading the computational burden between a pre-processing step and run-time.

Assumption 2 Additionally, we recognize the importance of accurately modeling the lower-frequency sound field to enhance the immersive experience in our applications (Brinkmann et al. 2019; Pind, Finnur 2020) and for the interaural time difference for source localization (Wightman et al. 1992).

Since realistic simulations of the lower frequency sound field are required and cannot be accurately modeled with GA, this work has focused on methods capturing the full sound field. This is typically done by approximating the acoustic wave equation by numerical methods. In the next section, we address the challenges of pre-calculating sound fields using numerical methods in the following section.

2.2 Limitations of numerical methods

Even when using efficient wave-based methods, the IRs must be calculated offline due to the computational requirements when real-time applications spanning a broad frequency range are considered. However, for dynamic, interactive scenes with numerous moving sources and receivers, the computation time and storage requirement for a lookup database become intractable (in the gigabytes range) since the IR is calculated for each source-receiver pair. Figure 2.1 shows the computation time for a $9\times3\times7$ m^3 room using our efficient in-house DG-FEM solver (Melander et al. 2020) with a spatial resolution of five points per wavelength (λ) using fourth-order polynomial basis functions running on an Nvidia V100 graphics processing unit (GPU). Due to the curse of dimensionality, the number of grid points grows cubically in three spatial dimensions with respect to spatial discretization (see Table 2.1), which is

Figure 2.1: Computation time with respect to frequency range when solving the wave-equation running the GPU accelerated DG-FEM solver (Melander et al. 2020) with 5 points per wavelength in a room of size 9 m × 3 m × 7 m = 189 m^3.

determined by the frequency range. This dependency between the grid resolution and the frequency range causes the calculation time to grow rapidly with frequency. The number of IRs required for all source and receiver combinations in a cubic domain with length L_{xyz} and the number of mesh points N can be calculated as

$$N = \left\lceil \frac{L_{xyz}}{\Delta x} \right\rceil^3, \tag{2.1a}$$

$$\texttt{\#IRs} = N^2 \quad , \tag{2.1b}$$

yielding the asymptotic growth of 6th order in the domain size $\mathcal{O}(L_{xyz}^6)$ as well as the spatial resolution $\mathcal{O}(\Delta x^{-6})$. This growth can quickly get intractable and becomes even more extensive when approaching the full audible frequency range. The storage requirement for the same room spanning frequencies up to 4 kHz is depicted in Table 2.2. We assume IRs of length 0.5s spanning frequencies up to $f_{\text{max}} = 1000$ Hz. When distributing the source and receiver positions in a uniform grid with a resolution at the Nyquist limit $\Delta x = 0.1715$ m, 4.7 million IRs are to be stored, requiring 9 GB of storage when 16-bit floating point precision is used. Using a coarser mesh discretized at $\Delta x = \lambda_{1000\text{Hz}} = 0.343$ m, 35,721 IRs are to be stored, requiring 612 MB of storage. The grid resolution is important for an accurate reconstruction of the IRs between mesh points when the source and receivers are allowed to move freely in the domain. Previous attempts to overcome the storage requirements of the IRs include work for lossy compression (Raghuvanshi et al. 2014), and lately, a novel portal search method has been proposed as a drop-in solution to pre-computed IRs to adapt to flexible scenes, e.g., when doors and windows are opened and closed (Raghuvanshi 2021).

	Grid nodes / m^3						
Δx	250 Hz	500 Hz	1,000 Hz	2,000 Hz	4,000 Hz	10,000 Hz	20,000 Hz
2 ppw	3	25	198	1,586	12,688	198,247	1.6M
6 ppw	84	669	5,353	42,821	342,571	5.4M	42.8G

Table 2.1: Number of mesh points per m^3 for a variety of frequencies for an optimal spatial resolution of ppw = 2 and an oversampled resolution of ppw = 6.

Δx	#locations	#IRs	storage
0.1715 m	2,173	4.7M	9 GB
0.343 m	567	321,489	613 MB

Table 2.2: Storage requirements for source/receiver pairs in a domain of size 9 m × 7 m, simulating frequencies up to 1 kHz for simulation time $t_{\text{max}} = 0.5$ s.

2.3 Data-driven methods to overcome limitations

To overcome the storage requirements intrinsic to the offline approach using numerical methods, a third category of methods is considered, namely scientific machine learning (SciML). SciML is an emerging and interdisciplinary field that combines machine learning with well-established theory from scientific computing and mathematical-physical modeling to tackle complex scientific problems. More accurate and comprehensive models can be developed by integrating machine learning methods with domain-specific knowledge about the underlying physical system. SciML has gained much interest recently (Bianco et al. 2019; Cai et al. 2021; Cuomo et al. 2022) with a wide range of applications. In this work, we have applied SciML in the form of deep neural networks to tackle the challenge of handling dynamic scenes with many moving sources and receivers, intending to remove the extensive storage requirement for discrete source/receiver locations in traditional methods. By learning a compact and efficient surrogate model interpolating in a grid-less domain that can execute in real-time, the limitations of pre-calculating and storing all combinations of source/receiver pairs can be overcome. A machine learning model learns the full wave field originating from the wave equation to intrinsically capture the wave phenomena required for an immersive reproduction.

Physics-informed neural networks (PINNs) (Psichogios et al. 1992; Lagaris et al. 1998; Raissi et al. 2019) are a class of machine learning techniques that combine physics-based knowledge with neural networks and can be used to solve differential equations. PINNs include the known physics during training and minimize their residual through the loss function and differ from traditional "black box" neural networks purely learning from data. It has recently seen a lot of attention, but the applications of PINNs in virtual acoustics are still limited (Borrel-Jensen et al. 2021; Moseley et al. 2020; Rasht-Behesht et al. 2021; Ma et al. 2023; Alkhadhr et al. 2021; Lee et al. 2023).

Another family of neural networks is *neural operators* designed to approximate mathematical operators contrary to functions (Z. Li et al. 2021; Lu et al. 2021; Cao et al. 2023). *DeepONet* is a specific neural operator architecture where the underlying theory stems from the universal operator approximation theorem (Chen et al. 1995), stating that a neural network (NN) with a single hidden layer of infinite width can approximate any nonlinear continuous functional or operator. Similar to PINNs, neural operators have seen a lot of attention in recent years. Still, applications in virtual acoustics are non-existing to the author's knowledge, although many promising applications in other fields are available, such as in fracture mechanics (Goswami, Yin, et al. 2022), diesel engine (Kumar et al. 2023), microstructure evolution (Oommen et al. 2022), bubble dynamics (Lin et al. 2021), bio-mechanics to detect aortic aneurysm (Goswami, D. S. Li, et al. 2022) and airfoil shape optimization (Shukla et al.

2023).

Both PINNs and DeepONets can learn a surrogate model that can be executed very efficiently at runtime (in the range of ms) and takes up little storage due to their intrinsic interpolation properties in grid-less domains; therefore, these methods have been investigated to overcome some of the limitations in traditional numerical methods. On the other hand, traditional numerical methods are useful for producing high-fidelity data for training the surrogate models.

A recent technique for handling parameter parameterization and model order reduction for accelerating numerical models is the reduced basis method (RBM) (Hesthaven et al. 2015; Llopis et al. 2022). Although very efficient, RBM cannot meet the runtime requirements regarding computation time for real-time virtual acoustics.

2.4 Data generation

2.4.1 Numerical methods - an overview

Since data-driven approaches often demand substantial amounts of data, this research has also focused on developing more efficient numerical techniques for wave propagation calculations while ensuring high accuracy in the solutions. Before delving into the details, let us briefly overview the current state-of-the-art numerical methods.

The most widely employed numerical method in acoustics is the *finite-difference time-domain method (FDTD)*. This method relies on spatial discretization using a uniform grid and approximates temporal and spatial derivatives through finite differences. Early applications of FDTD in acoustics were carried out by (Botteldoorena 1995; Savioja et al. 1994), and later, (Kowalczyk et al. 2011) introduced frequency-dependent boundary conditions as digital impedance filters. However, (Borrel-Jensen 2012) demonstrated experimentally that the frequency-dependent boundary formulation could become unstable when dealing with complex geometries. Higher-order schemes are also available (Van Mourik et al. 2014) but come with the expense of computation and implementation effort.

To address stability and flexibility concerns, a *finite-volume time-domain (FVTD)* formulation was developed, which also includes frequency-dependent boundary conditions (Hamilton 2016; Bilbao 2013; Bilbao et al. 2016; Chobeau et al. 2016). FVTD operates on unstructured meshes, providing greater geometrical flexibility compared to FDTD. One general limitation of FVTD is its inability to achieve higher-order accuracy, which can restrict its applicability in certain scenarios. When confined to a uniform grid, the FVTD formulation simplifies to the FDTD formulation for inner nodes yet retains a stable frequency-dependent boundary treatment. This formulation allows for highly efficient implementation on the GPU (Borrel-Jensen 2012; Webb et al. 2011), as no interdependencies are present when updating pressure values at spatial grid nodes. However, an inherent challenge when reducing the FVTD operating on a uniform grid arises from the need for *staircasing* approximation at the boundaries, leading to errors in the estimated error decay, even with a finely refined grid (Bilbao 2013).

The *finite-element method (FEM)* (Craggs 1994; Okuzono et al. 2018) and *spectral-element method (SEM)* (Pind et al. 2019) are volumetric discretization techniques, akin to the finite-

volume method, that offer exceptional geometric flexibility due to their support for unstructured meshes. These methods partition the domain into smaller subdomains called finite elements, wherein local basis functions are defined. The global solution is then expressed in terms of the local basis functions, obtained by minimizing the error residual through fitting weighted coefficients of test functions. The key distinction between FEM and SEM lies in their choice of basis functions. FEM is typically associated with 1st or 2nd-order polynomials, while SEM employs higher-order polynomial basis functions, which can lead to a faster error convergence rate of $\mathcal{O}(h^{N+1})$, with N being the polynomial order. Consequently, the superior accuracy of higher-order methods enables problem-solving using coarser discretization, resulting in more scalable and efficient solutions.

The *discontinuous Galerkin finite-element method (DG-FEM)* (Hesthaven et al. 2008) exhibits a close relationship with the SEM. Specifically, both methods share commonalities in defining local elements by applying identical basis functions, local matrix operators, and nodal sets. Nevertheless, a fundamental distinction arises in the computation of the global solution. In SEM, the solution is ascertained by employing global matrix operators, which are constructed by summing the local piece-wise basis functions to form global continuous basis functions. In contrast, DG-FEM adopts a different approach by not enforcing global continuity. Instead, computations take place locally within each element, with interactions between elements being managed through an interface flux procedure. This makes the method highly suitable for parallel computing (Melander et al. 2020).

The *boundary element method (BEM)* (Hargreaves et al. 2019) is founded on discretizing the Helmholtz equation in the frequency domain. Rather than solving the equation within the entire domain volume, the problem is transformed into an equivalent integral formulation restricted to the boundary. This transformation is achieved by representing the solution as a combination of fundamental solutions to the partial differential equation (PDE). One of the key advantages of this approach is its dimensionality reduction, as it only requires solving at the domain boundary. Additionally, BEM allows for straightforward modeling of exterior domains. However, it is important to note that BEM has its limitations. The system matrices involved in this method are dense (although small) compared to those encountered in volumetric methods, which tend to be sparse (but large). This density can lead to increased computational costs, particularly for larger problems. Despite this drawback, BEM remains a valuable technique for problems with specific geometries and boundary-focused phenomena.

The final method explored in this study is the *Fourier (or pseudo-spectral) method (FM)* (Canuto et al. 2006). This technique utilizes trigonometric basis functions and leverages the Fourier transformation to solve the partial differential equation by computing its derivatives in the frequency domain. The Fourier method is remarkably efficient, displaying spectral convergence and possessing the advantageous quality of avoiding dispersion errors commonly associated with traditional numerical methods. Nonetheless, the Fourier method does have certain limitations. It necessitates simple geometries and periodic or rigid boundary conditions, which restrict its applicability to specific problem domains. To address these limitations, previous works (Raghuvanshi et al. 2009a; Muñoz et al. 2017) have adopted a domain decomposition approach applying the Fourier method within rectangular domains while employing more flexible methods near the boundaries.

Part of this PhD research has been dedicated to investigating similar hybrid approaches,

which combine the strengths of different methods to tackle challenges associated with specific geometries and boundary conditions.

2.4.2 Domain decomposition

In the pursuit of developing more efficient numerical methods, domain decomposition techniques have been explored, and we have made the following assumptions in our investigations:

Assumption 3 In the domain of room acoustics, the predominant medium is air.

Assumption 4 The attenuation of sound in the air can be neglected.

Our investigations into domain decomposition techniques aim to improve efficiency by dividing the domain into simpler regions for the air partition while employing more complex partitions near the boundaries that adapt to the domain's geometry. By simplifying the geometries and neglecting air and boundary losses, we can utilize efficient (though less flexible) methods in the air partitions. Meanwhile, the partitions near the boundaries are handled by less efficient but flexible methods that accurately capture the geometry and boundary materials. The SEM stands out as a versatile approach that can handle complex geometries and frequency-independent and dependent boundaries (Pind et al. 2019). Its higher-order convergence properties and the efficient solution of the resulting system of equations due to sparsity in the matrices make it particularly well-suited for handling partitions near the boundaries. Our research has focused on two methods for simulating wave propagation in the air domain:

1. The wave-based method (WBM) was introduced in 1998 by (Desmet 1998) and employs trigonometric basis functions, which promise faster convergence than BEM and FEM, but with the drawback of being restricted to convex domains.

2. The Fourier method takes advantage of the known analytical solution to the wave equation in rectangular domains with simple boundary conditions. By using the Fast Fourier Transform and performing time-stepping in the Fourier domain with a spatial sampling resolution close to the Nyquist limit, the solution can be calculated very efficiently. When the signal is properly band-limited and sufficient modes are used, no errors are introduced. The drawback of this method is its restriction to rectangular domains with periodic/rigid boundary conditions.

However, coupling these air domain methods with the SEM near the boundaries presents a challenge. In the existing literature, the coupling of WBM has been accomplished with (lower-order) FEM using a weighted residual formulation to enforce interface conditions (van Hal and Hirschberg, 2005). Another approach was proposed by (Raghuvanshi et al. 2009a) by coupling the FM with the FDTD method using an FD scheme for interface handling, and in (Muñoz et al. 2017) the coupling was done with DG-FEM through an overlapping interface.

CHAPTER 3

Fast and accurate numerical methods for calculating sound propagation

As discussed in Chapter 1, various numerical methods exist for solving partial differential equations, and selecting an appropriate method for solving the wave equation depends on several factors:

Scalability. The convergence rate of a numerical method determines how quickly it approaches the true solution as the problem size increases. As indicated in Table 2.1, when expanding the frequency range/grid resolution or increasing the domain size, the number of grid nodes grows rapidly due to the curse of dimensionality. A higher convergence rate, preferably *spectral convergence* referring to an exponentially fast decay of errors (Canuto et al. 2006), is essential for efficient scalability. Methods with faster convergence rates are generally favored over those with linear or quadratic convergence.

Constant factor. The constant factor refers to the coefficient that multiplies the dominant term in the computational complexity, e.g., $\mathcal{O}(h^2)$ is asymptotically equivalent to a scaling of $\mathcal{O}(c \cdot h^2)$ with c the constant factor. Hence, it affects the actual time or memory consumed for a given problem size and significantly influences the efficiency of a numerical method. For instance, a method may exhibit spectral convergence but possess a large constant factor, resulting in a longer absolute computation time compared to a method with a slower convergence rate but a small, constant factor.

Parallelizability. Enhancing scalability involves parallelizing numerical methods to leverage multiple computations concurrently. Properties that facilitate parallelization include minimal data dependencies, allowing independent tasks to execute concurrently; limited communication overhead between parallel processes to prevent bottlenecks; and efficient data structures that scale well with problem size and parallelism. Efficient parallelization further contributes to the method's ability to effectively handle larger and more complex problems.

The numerical method's flexibility is also a crucial aspect to consider, particularly concerning the following:

Geometry. The method's ability to precisely capture the boundary shape is crucial in ensuring overall simulation accuracy for realistic domains. Some methods utilize staircase

approximations, while others handle non-uniform meshes, which can significantly impact the fidelity of the results.

Boundary conditions. Accurately modeling the boundary conditions to reflect the materials of the domain is essential. A robust numerical framework should exhibit the flexibility to handle a wide range of material types to accommodate various real-world scenarios.

Directionality. Auralization for interactive applications requires spatialization of the sound field to create an immersive experience. Many sound sources emit sound in specific directionality patterns, and receivers should be modeled to account for the characteristics of the human auditory system, i.e., modeled for two ears. This directional information is vital in achieving a more realistic and engaging auditory experience.

Certain crucial aspects should be considered when developing and evaluating numerical methods for accurate and efficient sound simulations. Previous research has demonstrated that both the SEM and DG-FEM possess several desirable properties (Hesthaven et al. 2008; Canuto et al. 2006), including:

- Spectral convergence,
- Sparse system matrices, enabling efficient solvers,
- Geometrical flexibility,
- Versatility in handling complex boundary conditions,
- Ability to accommodate arbitrary initial conditions/source functions, and
- Efficient parallelization for the DG-FEM scheme.

These advantageous properties make the SEM and DG-FEM well-suited for room acoustics simulations. However, these methods can still be computationally demanding despite possessing these attributes, as illustrated in Figure 2.1. To improve the efficiency, we have implemented domain decomposition techniques, dividing the domain into partitions encompassing the air, which can take the form of rectangular or convex shapes, along with arbitrary partitions near the boundaries. This approach takes into account the geometrical shape and boundary conditions of the domain. Specifically, the partitions near the boundaries apply the SEM to leverage its flexibility, while the partitions consisting of air utilize more efficient, albeit less flexible, methods. This partitioning strategy enables us to balance accuracy and computational efficiency in our sound simulations. In the context of air domains, our investigation has focused on two specific methods:

- WBM: This method employs trigonometric basis functions and is limited to convex domains.
- Fourier method: This approach exploits the known analytical solution to the wave equation and is limited to rectangular domains. Our primary interest lies in its coupling with SEM.

In the following sections, we will explain these methods and present our rationale for discontinuing further efforts in coupling the WBM with SEM. Instead, we will elaborate on why the FM has shown more promise for successful coupling with SEM, making it the preferred choice for our investigations.

3.1 The wave-based method

The WBM (Desmet 1998) is designed for solving steady-state problems described by the Helmholtz equation (a frequency domain version of the wave equation (1.1))

$$\nabla^2 p(\mathbf{x}) + k^2 p(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{3.1}$$

where ∇^2 is the Laplacian, $p(\mathbf{x})$ is the pressure field variable of the Helmholtz equation, $k = \omega/c$ is the wave number, c the speed of sound, $f(\mathbf{x})$ is the forcing function and Ω is the problem domain.

The procedure for solving the Helmholtz equation consists of four steps 1) partition the problem into convex subdomains, 2) selection of a suitable set of wave functions for each subdomain, 3) construct and solve the system of matrices yielding the wave function contribution factors, 4) postprocessing of the dynamic variables. The formulation of the residuals, the system of equations, and the postprocessing will not be explained here but is summarized in Report B, and more details can be found in (Van Genechten et al. 2012a; Pluymers 2006). A useful overview paper can be found in (Deckers et al. 2014). In the next section, we will explain the specific choices of basis functions and discuss the results for our convergence analysis when using a point source in a closed 2D domain.

3.1.1 Trigonometric wave functions

The steady-state fields $p(\mathbf{x})$ in each subdomain are approximated by a solution expansion $\hat{p}_w(\mathbf{x})$ in terms of the n_w number of wave functions ϕ_w for a α partition

$$p^{(\alpha)}(\mathbf{x}) \simeq \hat{p}^{(\alpha)}(\mathbf{x}) = \sum_{w=1}^{n_w^{(\alpha)}} p_w^{(\alpha)} \phi_w(\mathbf{x})^{(\alpha)} + \hat{p}_q^{(\alpha)}(\mathbf{x}) = \mathbf{\Phi}^{(\alpha)}(\mathbf{x}) \mathbf{p}_w^{(\alpha)} + \hat{p}_q^{(\alpha)}(\mathbf{x}), \tag{3.2}$$

where $\hat{p}_q^{(\alpha)}(\mathbf{x})$ represents a particular solution resulting from the forcing term in the Helmholtz equation, and $\mathbf{x}$ are the spatial coordinates. Each wave function $\phi(\mathbf{x})$ exactly satisfies the homogeneous part of the Helmholtz equation. There are two choices of wave functions for 2D bounded domains, the r- and s-set

$$\sum_{w=1}^{n_w^{(\alpha)}} p_w^{(\alpha)} \phi_w^{(\alpha)}(\mathbf{x}) = \sum_{w_r=1}^{n_{w_r}^{(\alpha)}} p_{w_r}^{(\alpha)} \phi_{w_r}^{(\alpha)}(\mathbf{x}) + \sum_{w_s=1}^{n_{w_s}^{(\alpha)}} p_{w_s}^{(\alpha)} \phi_{w_s}^{(\alpha)}(\mathbf{x}). \tag{3.3}$$

The wave functions are defined as

$$\phi_w^{(\alpha)}(\mathbf{x}) = \begin{cases} \phi_{w_r}^{(\alpha)}(x,y) = \cos(k_{xw_r}^{(\alpha)} x) e^{-jk_{yw_r}^{(\alpha)} y}, \\ \phi_{w_s}^{(\alpha)}(x,y) = e^{-jk_{xw_s}^{(\alpha)} x} \cos(k_{yw_s}^{(\alpha)} y). \end{cases} \tag{3.4}$$

The only requirement for Equation 3.4 to be an exact solution to the Helmholtz equation is that the wave numbers satisfy

$$(k_{xw_r}^{(\alpha)})^2 + (k_{yw_r}^{(\alpha)})^2 = (k_{xw_s}^{(\alpha)})^2 + (k_{yw_s}^{(\alpha)})^2 = k^2.$$

There are infinitely many solutions for the above relation, but the following have been proposed

$$(k_{xw_r}^{(\alpha)}, k_{yw_r}^{(\alpha)}) = \left(\frac{w_1^{(\alpha)}}{L_x^{(\alpha)}}, \pm\sqrt{k^2 - \left(k_{xw_r}^{(\alpha)}\right)^2} \right),$$
$$(k_{xw_s}^{(\alpha)}, k_{yw_s}^{(\alpha)}) = \left(\pm\sqrt{k^2 - \left(k_{yw_s}^{(\alpha)}\right)^2}, \frac{w_2^{(\alpha)}}{L_y^{(\alpha)}}, \right).$$

$L_x^{(\alpha)}$ and $L_y^{(\alpha)}$ are the dimensions of the smallest box surrounding the subdomain, $k = \omega/c$, ω being the angular frequency and $w_1 = w_2 = 0, 1, \ldots$. This choice of wave functions leads to standing and evanescent waves.

3.1.2 Convergence analysis

The convergence rate is examined for the steady-state solution at $f = 300$ Hz with point source location $s_{xy} = (1.2, 1.2)$ m in a 2 m$\times$2 m rectangular domain. In Figure 3.1, the convergence

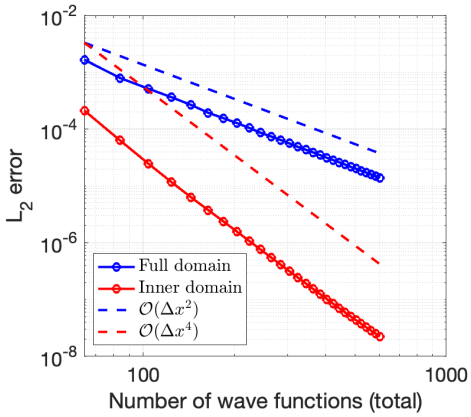


Figure 3.1: WBM convergence as a function of wave functions $Nw = Nw_r + Nw_s = 8, 28, 48, \ldots, 608$ in a 2 m $\times$ m domain at frequency $f = 300$ Hz. When excluding points on the boundary, the convergence is $\mathcal{O}(\Delta x^4)$ corresponding to the convergence of the SEM with third-order polynomial basis functions.

rate is summarized. We exclude points within a radius of $r = 0.4$ m from the singularity in the source point. The reference solution is constructed using the SEM on a very fine grid. WBM has a 4th-order convergence when calculating the L_2 error norm disregarding boundary collocation points, which corresponds to the convergence of an SEM with third-order polynomials as basis functions. Larger errors and slower 2nd-order convergence can be observed when including boundaries. Modeling pressure fields exited by point sources might be more challenging due to the abrupt injection, leading to highly oscillating waves. This requires more wave functions to capture the physics. A similar issue with degraded convergence has been observed when piston radiation from a boundary happens, resulting in discontinuous boundary conditions (Van Genechten et al. 2012b). The constant factor solving the smaller dense matrix systems resulting from the WBM compared to the larger sparse matrix systems resulting from the SEM/DG-FEM has not been investigated. From the above observations, it was decided to abandon further work in this direction since SEM and DG-FEM can obtain higher convergence rates and, therefore, might be a more efficient and scalable choice.

3.2 The hybrid Fourier/spectral element method

Domain decomposition is a powerful technique employed in solving partial differential equations (PDEs) by dividing the domain into multiple partitions that can be solved independently or in parallel. The solutions obtained in each partition are then combined at the interfaces by enforcing constraints to derive the overall solution to the PDE. One example of this approach is the hybrid FE-WBM, where domain decomposition is utilized to achieve the desired solution. The DG-FEM can also be perceived as a domain decomposition method, where fluxes constrain the solutions between the finite-element partitions. In this particular work, we have *not* adopted this approach, and we will explain the reasoning behind our method next.

The central idea in our work is to utilize the Fourier method for simulating the majority of the domain, which primarily consists of air, and then couple the Fourier method with the SEM in partitions near the boundaries. This method exhibits exceptional efficiency, with a time complexity of $\mathcal{O}(N \log N)$ for calculating the derivatives using the Fast Fourier Transform (FFT) for N degrees of freedom. Moreover, the method allows us to sample the spatial resolution at the Nyquist limit without introducing dispersion errors. However, this effectiveness comes with a limitation, as the method is confined to simple domains with periodic or rigid boundary conditions. To exploit the favorable features, the pressure fields are calculated independently for a single time step in each partition running either the Fourier method or SEM. This approach results in reflected wavefields at the interface, necessitating the implementation of interface handling to compensate for the reflected pressures in the partitions.

It is essential to note that this approach differs significantly from traditional domain decomposition methods. The unique characteristics of our domain decomposition technique introduce new challenges, particularly in handling interfaces, but they also offer promising opportunities for more efficient simulations.

The above-mentioned approach was first introduced in (Raghuvanshi et al. 2009b) and implemented for efficiency on the GPU in (Morales et al. 2015). The method is called *adaptive rectangular decomposition (ARD)* and consists of coupling the Fourier method with the

FDTD using 2nd and 6th order finite-differences for the interface handling. The research claims interface errors of −40 dB and 18×-24× speedups on the CPU using ARD compared to using FDTD in the full domain. The method is currently being used in Project Acoustics by Microsoft (Microsoft 2023). A related approach was taken by (Muñoz et al. 2017), coupling the Fourier method with the DG-FEM using an overlapping domain decomposition method for interface handling. Overlapping domains are imposed due to the requirements of including a Gaussian window function to impose periodicity on the Fourier method for interface handling. The method claims low errors and stable long-time simulations; however, no information is available concerning the method's efficiency.

We have investigated methods to couple the Fourier method with the SEM using finite-difference schemes inspired by ARD.

3.2.1 Paper D: Interface handling

Additional details about spatiotemporal interpolation can be found in Report A (excluded from Paper D due to length limitations). The main contribution of this work is the coupling between FM and SEM; in this presentation, we will further discuss the challenges.

This method aims to accurately and efficiently solve the wave equation from Equation 1.1. We can formulate a unified framework for communicating pressures between partitions running *any* method following the same approach as (Raghuvanshi et al. 2009b). The boundary condition is imposed at the pressure node N. Without lack of generality, assume that two independent second-order (2,2) FDTD update schemes are running in each partition and denote the pressures in partition 1 as $p_{1,i}$ and partition 2 as $p_{2,i}$ with subscripts denoting the partition and corresponding node index, respectively. Δx and Δt are the spatial and temporal resolutions, respectively, and the superscript $p^{(n)}$ denotes the discretized time steps. Then, the interface communication can be handled as follows:

1. Calculate the pressures in each domain as completely independent partitions with Neumann boundary condition at the interface

$$\begin{aligned} p_{1,N}^{(n+1)} &\leftarrow \frac{c^2\Delta t_1^2}{\Delta x_1^2}\left(2p_{1,N-1}^{(n)} - 2p_{1,N}^{(n)}\right) + 2p_{1,N}^{(n)} - p_{1,N}^{(n-1)} + F_{1,N}^{(n)}, \\ p_{2,1}^{(n+1)} &\leftarrow \frac{c^2\Delta t_2^2}{\Delta x_2^2}\left(-2p_{2,1}^{(n)} + 2p_{2,2}^{(n)}\right) + 2p_{2,1}^{(n)} - p_{2,1}^{(n-1)} + F_{2,1}^{(n)}. \end{aligned} \tag{3.5}$$

2. Remove the residual part at time n corresponding to the reflected pressures

$$p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} - \frac{c^2\Delta t_1^2}{\Delta x_1^2}\left(p_{1,N-1}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} - \frac{c^2\Delta t_2^2}{\Delta x_2^2}\left(p_{2,2}^{(n)}\right). \tag{3.6}$$

3. Transfer the removed residual part at time n to the neighboring partition(s)

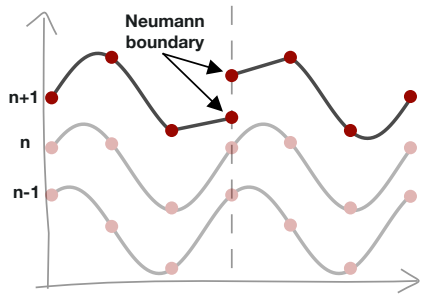
$$p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} + \frac{c^2\Delta t_1^2}{\Delta x_1^2} \cdot \left(p_{2,2}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} + \frac{c^2\Delta t_2^2}{\Delta x_2^2} \left(p_{1,N-1}^{(n)}\right). \tag{3.7}$$

In Figure 3.2, the intermediate pressure field is depicted after the Neumann boundary condition has been imposed at the interface after a time step Δt before interface handling. This is a non-physical state solely due to the procedure of calculating the pressure fields separately in the partition and afterward compensating for the reflected pressures. This introduces non-smooth second derivatives known as 'shocks' in the literature, and it is well-known to cause challenges for SEM and DG-FEM. If shocks are not handled accordingly, the scheme introduces large errors and could get unstable (although not observed here). An investigation has been made by comparing the Laplacian term in the wave equation for the SEM and FDTD methods

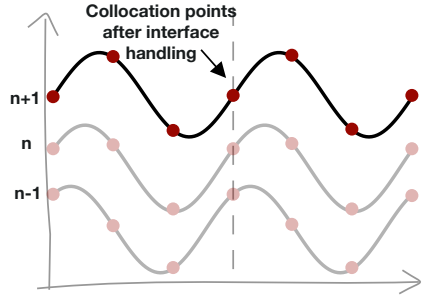
$$\text{(SEM)} \quad \mathbf{P}^{(n+1)} = 2\mathbf{P}^{(n)} - \mathbf{P}^{(n-1)} \overbrace{\boxed{-c^2\Delta t^2 \mathcal{M}^{-1}\left(\mathcal{S}\mathbf{P}^{(n)}\right)}}^{\text{SEM Laplacian}} + \Delta t^2 \mathbf{F}^{(n)}, \tag{3.8}$$

$$\text{(FDTD)} \quad \mathbf{P}^{(n+1)} = 2\mathbf{P}^{(n)} - \mathbf{P}^{(n-1)} \overbrace{\boxed{+\frac{c^2\Delta t^2}{\Delta x^2} K\mathbf{P}^{(n)}}}^{\text{FDTD Laplacian}} + \Delta t^2 \mathbf{F}^{(n)}, \tag{3.9}$$

where $\mathcal{M}$ is the mass matrix and $\mathcal{S}$ is the stiffness matrix (Hesthaven et al. 2008). The Laplacians annotated with boxes above are plotted in Figure 3.3 for the second-order interface scheme applied to the second-order FDTD method and the first and second-order SEM. The two methods run simultaneously in the entire domain (no coupling) with the residual subtracted corresponding to applying only Equation 3.5 and Equation 3.6, and leave out Equation 3.7. This corresponds to transforming Neumann to Dirichlet boundary conditions with

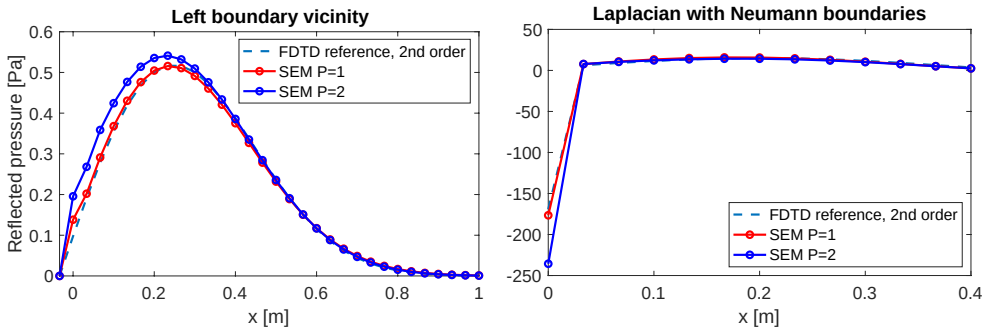


(a) Calculate time step $n+1$ from time steps n and $n-1$ with homogenous Neumann boundary conditions at interface.



(b) Adjust pressures near the interface at time $n+1$.

Figure 3.2: Illustration of the interface handling with the same spatial and temporal resolutions in both partitions.



(a) The left-traveling wave reflects at a left Neumann boundary when subtracting the interface residual at each time step. We notice much bigger errors for higher-order SEM than the first-order SEM and the FDTD method.

(b) The Laplacian term imposing Neumann boundaries introduces a shock, causing the second-order derivatives to be non-smooth when subtracting the interface residual at each time step. We notice much bigger errors for higher-order SEM than the first-order SEM and the FDTD method.

Figure 3.3: Accuracy of the Laplacians for the FDTD scheme and SEM when the interface residual is subtracted.

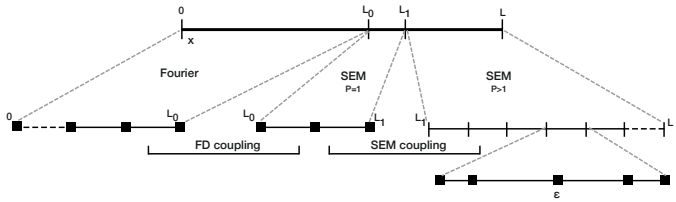


Figure 3.4: Domain decomposition coupling FM with SEM by introducing a first-order SEM layer to stabilize the scheme.

endpoints $p(-\Delta x, t) = p(l + \Delta x, t) = 0$ outside the domain. No 'interface' errors are introduced in the FDTD simulation since the interface scheme, and the simulation scheme have the same order. We note that much bigger errors are introduced for the higher-order SEM due to the shock introduced when enforcing Neumann boundaries at each time step, which can be observed both in the pressure plot and the plot of the Laplacians at $x = 0$. A simple remedy was to add an SEM layer of first-order polynomials near the interface, as illustrated in Figure Figure 3.4, where the number of nodes in the layer should correspond to half the interface scheme order. However, this comes with the disadvantage of reducing the overall convergence to the polynomial order at the interface, acting as a bottleneck for the whole solution. In (Hesthaven et al. 2008) [Section 5.6], filtering methods have been proposed to reduce the errors in the presence of shocks, and it should be investigated if this method can be applied to reduce the approximation errors for higher-order SEMs.

3.3 Contributions

In Paper D and the complementary material in Report A, we examine the domain decomposition method suggested in (Raghuvanshi et al. 2009b) in a new setting, where the Fourier method is coupled with the SEM.

- An interface coupling between partitions running the Fourier method and the SEM is proposed. The interface coupling is based on the FDTD scheme defined on a uniform grid similar to (Raghuvanshi et al. 2009b) with the key difference of coupling the Fourier method with SEM instead of coupling the Fourier method with FDTD.

- The proposed Fourier-SEM coupling alleviates the need for oversampling in the majority of the domain corresponding to air and, moreover, exploits the Fast Fourier Transform for efficiently solving the wave equation in the Fourier domain. Overall, a $18\times$ speedup was reported in a 1D domain where the Fourier method was applied in 95% of the domain corresponding to the air partition while achieving relative errors below 10% with interface errors around -36 dB.

- An interpolation procedure ensuring small interface errors was carefully explained and proposed. To exploit the efficiency of the Fourier method capable of operating close to the spatial Nyquist limit, interpolation in the temporal and spatial dimensions is required and has not been described in the literature. Report A describes the interpolation in greater details.

- A remedy for handling shocks due to the unphysical coupling was proposed for improving the accuracy of the approximation of the Laplacian in the SEM. The remedy was to add a layer of 1st order polynomial basis functions between the SEM partition and the interface, drastically improving the accuracy. However, this affects the global error convergence of the SEM, effectively reducing the convergence to 1st order.

CHAPTER 4

Handling dynamic scenes with moving sources and receivers

Addressing interactive dynamic scenes with moving sources and receivers poses a challenging problem for traditional numerical methods. Therefore, we have focused on building surrogate models that should be capable in real time to predict the IR for any source/receiver pair in a grid-less domain. Deep neural networks have emerged as a powerful tool for solving partial differential equations by leveraging their ability to learn complex patterns. The fundamental concept involves approximating the solution of the equation (or the operator) by training the input/output relation through the minimization of a loss function. The network is trained using pairs of input data, including spatial and temporal information and the corresponding solution data. Its objective is to generalize this learning process, enabling the prediction of solutions for unseen inputs with high accuracy. Moreover, deep neural networks can provide solutions much more efficiently than traditional methods once trained by exploiting optimized hardware, such as GPUs.

In this work, we have focused on two main techniques: physics-informed neural networks (PINNs) and neural operators using the DeepONet framework. PINNs were introduced in 2017 and later published in 2019 (Raissi et al. 2019). However, the idea of constrained neural networks can be traced back to earlier works such as (Psichogios et al. 1992) and (Lagaris et al. 1998). One of the first instances that can be considered a PINN was presented in (Dissanayake et al. 1994), where a quasi-Newtonian approach was used, and gradients were evaluated using finite-differences. It is important to note that while PINNs are a notable framework for solving PDEs with neural networks, they are not the only one, and an extensive review of various approaches can be found in (Cuomo et al. 2022). The DeepONet framework (Lu et al. 2021) is a specific realization of neural operators. Unlike function regression, operator regression aims to learn the mapping from one function space (inputs) to another function space (output), where the learned operator can be evaluated at arbitrary (continuous) locations.

In the following, we will introduce general concepts related to deep neural networks and dive into the methods in more detail.

4.1 Deep neural networks

Deep learning is a subfield of machine learning that revolves around artificial neural networks with multiple layers. Its diverse applications include image classification, computer vision, speech recognition, language translation, autonomous driving, bioinformatics, and more. More recently, deep learning has found its way into scientific domains, demonstrat-

ing its utility in earthquake detection, fluid mechanics, turbulent modeling, and more. The widespread adoption of deep learning is facilitated by accessible software packages such as TensorFlow, PyTorch, and Jax, as well as advancements in hardware technology, such as GPUs and TPUs, which significantly enhance its computational efficiency.

4.1.1 A short introduction

The Feed-Forward Neural Network (FNN) (Goodfellow et al. 2016) is the simplest neural network and is used as building blocks for more advanced networks. An FNN is depicted in Figure 4.1 and consists of an input layer $\mathbf{x}$, L hidden layers, and an output layer and maps an input $\mathbf{x}$ to an output $\mathbf{y}$ as

$$\mathbf{y} = (g_1 \circ \ldots \circ g_L)(\mathbf{x}), \quad \text{where} \tag{4.1a}$$

$$g_i(\mathbf{x}) = \sigma_i(\mathbf{W}^i \mathbf{x} + \mathbf{b}^i), \tag{4.1b}$$

and $\sigma_i(\mathbf{x})$ is a non-linear activation function, except for the last layer applying the identity mapping $\sigma_L(\mathbf{x}) = \mathbf{x}$. A multilayer perceptron (MLP) is a special case of an FNN, where every layer is fully connected, and the number of nodes in each layer is the same.

The weights $\mathbf{W}^i$ and biases $\mathbf{b}^i$ are the parameters to learn. Hence, each layer receives information from the previous layer and passes it forward to the next layer after a combination of scaling determines the weights W, shifting determined from the bias b, and a non-linear mapping by the activation function σ. The weights $\mathbf{W^i}$ and biases $\mathbf{b^i}$, $\forall i = \{1, \ldots, L\}$ are identified through a minimization problem

$$\arg\min_{\mathbf{W},\mathbf{b}} \mathcal{L}(\mathbf{W}, \mathbf{b}) = \| f(\mathbf{x}^*) - \hat{f}(\mathbf{x}^*) \| = \| f(\mathbf{x}^*) - \mathcal{N}(\mathbf{x}^*; \mathbf{W}, \mathbf{b}) \| \tag{4.2}$$

where $\mathbf{x}^*$ are the discrete collocation points and $\| \cdot \|$ is the mean squared norm used in this work.

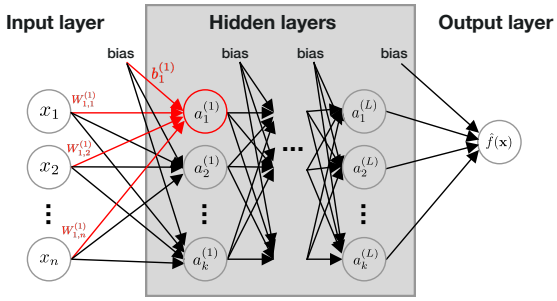


Figure 4.1: Feed-Forward Neural Network architecture with n inputs, L hidden layers with k neurons and one output layer. The weights and connections highlighted in red contribute to the output of the highlighted neuron in the first hidden layer, i.e., $a_1^{(1)} = \sum_{i=1}^{n} W_{1,i}^{(1)} x_i + b_1^{(1)}$ and similar for all neurons in the first hidden layers written in matrix form $\mathbf{a}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$.

Typical, the loss is minimized through gradient descent by taking a step γ_n in the opposite direction of the gradient as

$$\mathbf{x_{n+1}} = \mathbf{x_n} - \gamma_n \Delta \mathcal{L}(\mathbf{x}_n), \tag{4.3}$$

where $\Delta(\cdot)$ is the differential operator and γ_n is the step size. ADAM (Goodfellow et al. 2016) was used in this work, which includes a variate learning rate and momentum, helping the optimizer escape local minima. The loss function is minimized in terms of the network parameters and therefore is very high-dimensional. To efficiently calculate the gradient, the back-propagation algorithm (Rumelhart et al. 1986) is used to dynamically calculate the gradient of each of the network weights and biases

$$\frac{\partial \mathcal{L}}{\partial W^l_{jk}}, \quad \frac{\partial \mathcal{L}}{\partial b^l_j}, \tag{4.4}$$

by applying the chain rule. The subscript jk denotes the weight for the connection from node k in layer $l-1$ to node j in layer l. An intuitive explanation can be found in (Nielsen 2017).

Typical activation functions σ are the ReLu $\hat{x} \mapsto \hat{x}^+$, sigmoid $\hat{x} \mapsto 1/(1+e^{\hat{x}})$ and tanh $\hat{x} \mapsto (e^{\hat{x}} - e^{\hat{-x}})/(e^{\hat{x}} + e^{\hat{-x}})$ functions, but in this work the sine function greatly outperformed the other choice as will be discussed in subsection 4.1.3.

4.1.2 Overcoming the curse of dimensionality

The curse of dimensionality refers to the exponential increase in data size and computational complexity when the number of dimensions in the datasets grows. Numerical methods are subject to the curse of dimensionality because their computational complexity increases exponentially with the number of dimensions, as also discussed in Chapter 1. As an example, let us represent a function $p(x)$ approximated by polynomial basis functions in terms of a series expansion $\hat{p}(x)$ in 1D as

$$\hat{p}(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots, \tag{4.5}$$

where a_i are the coefficients to be determined. Now, representing a function $p(x, y)$ in 2D, we would require the basis functions to span a 2D polynomial space $\mathbb{R} \times \mathbb{R}$ of order P as (Hesthaven et al. 2008)

$$\mathcal{P}^P = \text{span}\{x^\alpha y^\beta\}, \quad \alpha, \beta \geq 0, \alpha + \beta \leq P. \tag{4.6}$$

The necessary terms in the series expansion can be deduced from Pascal's triangle of order P corresponding to the binomial expansion $(x+y)^P$

$$\begin{array}{lccccccccc} & & & & & a_0 & & & & \\ & & & & +a_1 x & & +a_2 y & & & \\ \hat{p}(x,y) = & & & +a_3 x^2 & & +a_4 xy & & +a_5 y^2 & & \\ & & +a_6 x^3 & & +a_7 x^2 y & & +a_8 x y^2 & & +a_9 y^3 & \\ & \dots & & \dots & & \dots & & \dots & & \dots \end{array} \tag{4.7}$$

The series expansion in 3D written in general form for basis functions ϕ is

$$\hat{p}(x, y, z) = \sum_{i}^{P} \sum_{j}^{P} \sum_{k}^{P} a_{i,j,k} \phi_i(x) \phi_j(y) \phi_k(z), \tag{4.8}$$

where ϕ could be, for example, Legendre or Lagrange polynomials typical for SEM or trigonometric functions typical for Fourier series. Also, Taylor expansions or Monte Carlo sampling suffer from the same limitations.

Techniques exist to overcome the curse of dimensionality, such as Principal Component Analysis, Proper Orthogonal Decomposition, and Reduced Basis Methods (Llopis et al. 2022), aiming to reduce the complexity of high-dimensional systems while preserving their essential behavior. Of particular interest for our work are deep neural networks. Deep neural networks are inherently capable of learning hierarchical representations from high-dimensional data. This is possible due to the multiple layers of non-linear transformation enabling the network to learn features and patterns, effectively reducing the problem's dimensionality and mitigating the curse of dimensionality. A theoretical investigation for DeepONets can be found in (Lanthaler et al. 2022) but is out of scope for this thesis work. This is an important property essential for the scalability of neural networks, including DeepONets.

4.1.3 Spectral bias

It is well-known that deep neural networks first learn the lower frequency modes of the data and suffer from learning the higher frequency modes. This phenomenon is known as spectral bias (Rahaman et al. 2018; Basri et al. 2020) and can be observed in Figure 4.2, where the 1D wave equation is learned using `tanh` activation functions lacking the high-frequency content of the signal. This problem can be addressed by passing the temporal and spatial coordinates through a Fourier feature mapping that enables a deep FNN to learn the high-frequency modes of the data shown in Figure 4.2(b). The Fourier mapping can be written

$$\gamma(\mathbf{x}) = \begin{pmatrix} \cos(2\pi \mathbf{B}\mathbf{x}) \\ \sin(2\pi \mathbf{B}\mathbf{x}) \end{pmatrix} \quad \text{for } \mathbf{B} \in \mathbb{R}^{m \times d}, \tag{4.9}$$

where m is the number of frequencies and $\mathbf{B}$ is the Fourier mapping matrix. The Fourier mapping matrix can be defined in various ways, e.g., as a Gaussian mapping or a positional encoding mapping as a diagonal matrix, as investigated in Paper C. Using the `sine` activation function in all layers is depicted in Figure 4.2(c) and shows even better results compared to applying feature expansion techniques. The predictions were all made using DeepONet with the standard MLP network architecture and show RMSE[1] errors of 0.12 Pa, 0.003 Pa, and 0.001 Pa for a), b), and c), respectively. In (Benbarka et al. 2022) (section 3.1), it has been shown that passing the temporal and spatial coordinates through a Fourier feature mapping is equivalent to a Fourier series and we will briefly summarize the findings in the following. A

[1] $\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} (p_{\text{ref}_i} - p_{\text{pred}_i})^2}{N}}$

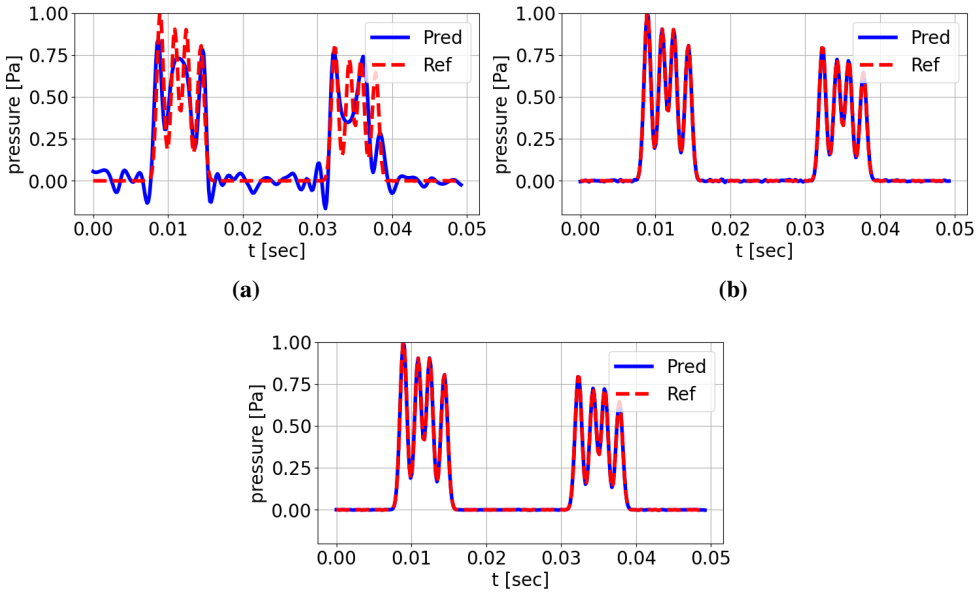


Figure 4.2: Examples showing the spectral bias when learning a 1D wave propagation problem for different activation functions and the impact using Fourier feature expansion techniques. (a) `tanh` activation function (RMSE = 0.12 Pa), (b) Fourier feature mapping with the `tanh` activation function (RMSE = 0.003 Pa), and (c) `sine` activation function, no Fourier feature mapping (RMSE = 0.001 Pa). (b) and (c) clearly outperforms (a), struggling to capture the high-frequency content.

Fourier series is a weighted sum of sines and cosines with incrementally increasing frequencies that can reconstruct any periodic function using enough frequencies and can be written (here assuming the input is periodic over the bound of the input)

$$f(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{N} \times \mathbb{Z}^{d-1}} a_{\mathbf{n}} \cos(2\pi \mathbf{n} \cdot \mathbf{x}) + b_{\mathbf{n}} \sin(2\pi \mathbf{n} \cdot \mathbf{x}), \tag{4.10}$$

where $a_{\mathbf{n}}$ and $b_{\mathbf{n}}$ are the Fourier coefficients. Writing the above equation in vector form gives

$$f(\mathbf{x}) = \left((a_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}, (b_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}} \right) \cdot \begin{pmatrix} \cos(2\pi \mathbf{B}' \mathbf{x}) \\ \sin(2\pi \mathbf{B}' \mathbf{x}) \end{pmatrix}. \tag{4.11}$$

If we let the activation function be the identity function, we get

$$y(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \mathbf{W}\gamma + \mathbf{b}, \tag{4.12}$$

and comparing with Equation 4.11 we find that $\left((a_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}}, (b_{\mathbf{n}})_{\mathbf{n} \in \mathbf{B}} \right)$ is equivalent to $\mathbf{W}, \mathbf{b} = 0$, and $\mathbf{B}' = \mathbb{N} \times \mathbb{Z}^{d-1}$ takes all possible combinations of $\mathbf{n}$ as defined in Equation 4.10.

A comparable remedy to address the spectral bias is to apply the `sine` activation function, showing superior performance for wave propagation problems compared to the traditionally used activation functions such as `tanh` and `relu` depicted in Figure 4.2(c). (Benbarka et al. 2022) also showed that a Fourier-mapped perception is structurally similar to applying `sine` activation function in the first layer.

Proper initialization of the weights in a deep neural network is important to ensure the variance of the activations remains the same across different layers. Using the standard initializations (Glorot et al. 2010; He et al. 2015) for the `sine` activation is not optimal. Instead, a strategy was proposed in (Sitzmann et al. 2020) (section 3.2) with the key idea of preserving the distribution of activations through the network, not dependent on the number of layers. We have initialized the weights of the networks as

$$w_i \sim \mathcal{U}\left(-\frac{\sqrt{6/n}}{k}, \frac{\sqrt{6/n}}{k}\right), \tag{4.13}$$

where n denotes the number of input neurons to the ith neuron. The first layer is initialized with an angular frequency w_0 such that the sine functions $\sin(w_0 \cdot \mathbf{Wx} + \mathbf{b})$ spans multiple periods.

4.2 Physics-informed neural networks

PINNs (Raissi et al. 2019) can solve differential equations in their general form

$$\mathcal{F}(u(\xi); \gamma) = f(\xi), \quad \xi \in \Omega, \tag{4.14a}$$

$$\mathcal{B}(u(\xi)) = g(\xi), \quad \xi \in \partial\Omega, \tag{4.14b}$$

defined on the domain $\Omega \in \mathcal{R}^N$, $N \in \mathbb{Z}$, where the location is denoted by $\xi = [x_1, x_2, \dots, t]$ with x and t denoting spatial and temporal dimensions, respectively, u represents the unknown field to be solved for, γ are the parameters of the physics in the differential equation, f is a forcing function, $\mathcal{F}$ is a (non-linear) differential operator, $\mathcal{B}$ is an operator denoting initial/boundary conditions, g is the initial/boundary function. A deep neural network $\mathcal{NN}(\xi; \Theta)$ must learn the approximation $\hat{u}(\xi) \approx u(\xi)$ by finding the optimal network parameters Θ by minimizing a loss $\mathcal{L}$ typically through gradient descent. In PINNs, the loss function is a weighted sum of the differential equation and the coupled initial/boundary conditions

$$\mathcal{L} = \lambda_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\Theta) + \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}}(\Theta) + \lambda_{\text{data}}\mathcal{L}_{\text{data}}(\Theta), \tag{4.15}$$

and the optimization problem can be stated as

$$\Theta^* = \arg\min_{\Theta}(\mathcal{L}). \tag{4.16}$$

Note that arbitrarily many losses can be added, and the data loss could be absent for purely non-data-driven approaches. An excellent outlook of PINNs is given in (Cuomo et al. 2022).

4.2.1 Universal function approximation theorem

Typical for most deep learning techniques is that they are based on the assumption of function approximations of neural networks (Hornik et al. 1989). The Universal Approximation Theorem states that a feedforward neural network with a single hidden layer containing a sufficient number of neurons or units can approximate any continuous function to an arbitrary degree of accuracy within a compact input domain.

Formally, let $f : \mathbb{R}^n \to \mathbb{R}^m$ be the arbitrary function to be approximated by a network, then the Approximation Theorem states that for any $\epsilon > 0$ and any continuous function f, there exists a neural network F with a single hidden layer $F(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ such that

$$|f(\mathbf{x}) - F(\mathbf{x})| < \epsilon, \tag{4.17}$$

where $\mathbf{W} \in \mathbb{R}^m \times \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. The Universal Approximation Theorem provides a theoretical foundation that the network with arbitrarily many neurons can form complex combinations of simple functions to approximate any continuous function bounded by an arbitrary small ϵ.

The theorem does not provide any number of neurons or the specifics of the network architecture to achieve a given accuracy. The Universal Approximation Theorem only guarantees a small *approximation* errors for a sufficiently large network but does not consider *optimization* or *generalization* errors. The three main categories of errors are listed below

Approximation errors

Bias error (or underfitting) occurs when the network fails to capture the underlying pattern and complexity of the problem to learn. The consequences are high training errors and poor generalization of unseen data. Bias errors could indicate that the network's capacity or complexity is inadequate for the problem but could also be caused by inadequate hyperparameters or issues related to the optimizer.

Irreducible error represents noise or randomness in the data that cannot be reduced in the model and can be attributed to measurement data or some inherent variability in the data. The irreducible error could be handled by adjusting the loss metric or adding regularizer terms.

Optimization errors arises when the model is unable to converge to an optimal solution caused by bad hyperparameter tuning (e.g., inadequate learning rate), vanishing or exploding gradient (e.g., bad parameter initialization), or getting stuck in local minima (e.g., when using optimizers with fixed learning rates). Proper hyperparameter tuning and choosing more sophisticated optimizers could mitigate these issues.

Generalization errors

Variance error (or overfitting) occurs when the network becomes too sensible to the training data and fails to generalize to unseen data. Instead of *learning* the underlying patterns, it *memorizes* the data leading to low training errors but high validation/test errors. Variance errors typically indicate that the network is too complicated or the training data is too sparse.

Hence, setting up a deep neural network that approximates and generalizes well to the problem of interest requires investigating and mitigating the three categories of errors.

4.2.2 Paper A: Solving the wave equation with PINNs

A PINN method in one dimension was developed, which learns a compact and efficient surrogate model with parameterized moving sources and impedance boundaries and satisfies a system of coupled equations. A fully data-free approach was taken, where only the underlying physics are included in the training and their residual minimized through the loss function. No data was included to train the model, allowing insights into how well PINNs predict sound fields in acoustic conditions. The Gaussian impulse was used as the initial condition tested with frequency-independent and dependent impedance boundaries, and the setup is depicted in Figure 4.3. Including only information about the underlying physics, the governing partial differential equation (PDE) and initial conditions (ICs) can be learned by minimizing the loss $\mathcal{L}$

$$\arg\min_{\mathbf{W},\mathbf{b}} \mathcal{L}(\mathbf{W},\mathbf{b}) = \mathcal{L}_{\text{PDE}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{ADE}}, \tag{4.18}$$

where $\mathcal{L}_{\text{PDE}}$, $\mathcal{L}_{\text{IC}}$, $\mathcal{L}_{\text{BC}}$, and $\mathcal{L}_{\text{ADE}}$ are the losses for the PDE, the initial condition, the boundary conditions, and the coupled auxiliary equations used for approximating frequency-dependent boundaries (Troian et al. 2017), respectively. The losses might have individual scaling; therefore, the weights $\lambda_\bullet$ have been added to balance the losses (weights for $\mathcal{L}_{\text{ADE}}$ are included in the loss function itself). We exploit the automatic differentiation leveraged by modern soft-

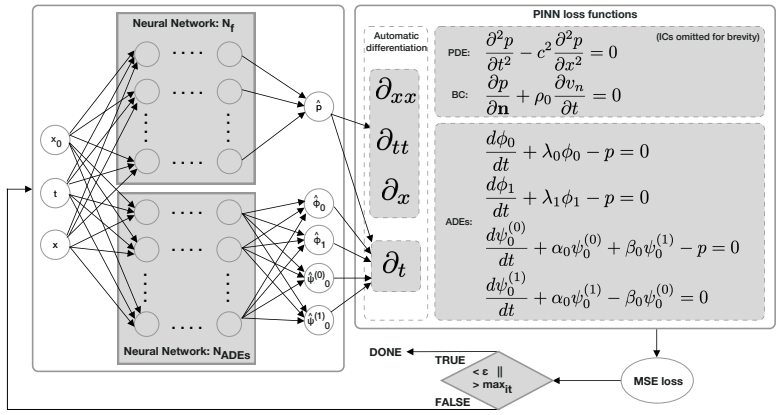


Figure 4.3: PINN scheme for frequency-dependent boundaries. Left: Two fully connected Feed-Forward Neural Network architectures, $\mathcal{N}_f$ (PDE + ICs) and $\mathcal{N}_{\text{ADE}}$ (ADEs). Right: The governing physical equations and ADEs are coupled via the loss function (ICs and scaling terms are omitted for brevity). Training is done when a maximum number of epochs is reached or the total loss is smaller than a given threshold.

ware packages to evaluate the derivatives efficiently. For example, the PDE can be evaluated as follows using Tensorflow through the SciANN (Haghighat et al. 2021) framework as

```
L = diff(p,x,order=2) - c^2*diff(p,t,order=2).
```

During the process of developing a PINN setup for the coupled wave equation for modeling frequency-dependent boundaries, several learnings were found to have a huge impact on the convergence and solution accuracy:

The loss components may have their own scale leading to difficulties when minimizing the total loss function. For example, the neural network might be able to approximate the initial conditions but not the boundary conditions. It could also respect both initial and boundary conditions but not the PDE. In this work, the weights were determined manually. Be aware that the loss $\mathcal{L}_{\text{ADE}}$ also consists of several weighted losses, one for each accumulator.

Normalization of the auxilary outputs ϕ_0, ϕ_0, $\psi_0^{(0)}$ and $\psi_0^{(1)}$ in the loss $\mathcal{L}_{\text{ADE}}$. The magnitude of these quantities takes very small values; moreover, they vary by orders of magnitude independently, causing troubles for the optimizer to find proper optima if not normalized. The normalized values should only be used in the loss and not when calculating the pressure at the boundary using the normal velocity u_n.

Activation choice. Using `sine` for the network $\mathcal{N}_f$ activations predicting the pressures outperformed the traditional choices.

Distribution of data samples with almost half of the samples located at the boundaries and a quarter of the samples at the initial condition depicted in Figure 4.4. Without this distribution, the BC and IC errors were never learned, and the errors were propagated through the whole solution of the domain.

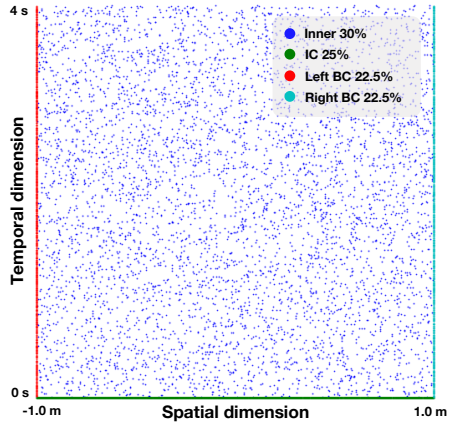


Figure 4.4: Grid distribution between inner points (30%), initial (25%) and boundary (45%) points in a 1D domain of dimension $x \in [-1.0, 1.0]$ m. The temporal dimension is $T \in [0, 4]$ s with normalized speed of sound $c = 1$ m/s corresponding to $T_{\text{phys}} = [0.0, 0.012]$ s for $c_{\text{phys}} = 343$ m/s.

4.2.3 State-of-the-art techniques

The development and use of PINNs is an active research area, and novel techniques for improving their performance have appeared around and after the publication of this research, and we will do a short review of the most relevant research.

The loss weights have been tuned by hand for this work, but these might not have been the most optimal choices, and moreover, it is painful to find the right combination of weights. Adaptive techniques for tuning the loss weights, referred to as dynamic loss weighting, have been proposed in (S. Wang, Yu, et al. 2022; X. Jin et al. 2021; Bischof et al. 2021), and the idea is to turn the loss weights into trainable parameters combined into a (separate) optimizer updated according to some heuristic.

Like dynamic weights, self-adaptive weights scale the different contributions, but instead of balancing the loss terms, the individual samples in each loss term are weighted. Using point-wise weights (McClenny et al. 2023), the loss function can be minimized w.r.t. the network parameters but maximized w.r.t. the point-wise loss weights. Applying self-adaptive weights, the points with large residuals will be more important than those with smaller residuals.

The collocation points can be distributed in several ways. This work used a static Latin hypercube sampling for the inner domain, with a relatively large portion of the overall samples located at the IC and BCs. (Wu et al. 2023) has explored the use of resampling and residual-based adaptive refinement techniques. Another recent paper titled 'Respecting causality is all you need for training physics-informed neural networks' by (S. Wang, Sankaran, et al. 2022) has gotten much attention. Because we usually train PINNs using a set of collocation points and try to minimize the PDE residual by giving equal importance to each point, we may not respect physical causality. The authors in (S. Wang, Sankaran, et al. 2022; Daw et al. 2023) propose overcoming this problem by sampling and weighting techniques to propagate the solution following its 'preferential direction of propagation.' For the wave propagation problem, that could be to propagate the initial conditions first by giving more importance to earlier times at earlier training steps and then gradually giving more importance to the later times after more training iterations.

4.3 Neural operators

Neural operators represent a class of neural network architectures designed to directly solve partial differential equations (PDEs) and other mathematical operators. Unlike traditional function learning typically done in deep learning, neural operators aim to learn mappings between infinite-dimensional Banach spaces. This methodology effectively captures the mapping from an input space of functions to an output space of functions, offering a generalized solution for a parametrized PDE.

Among the most well-known neural operators are the DeepONet (Lu et al. 2021) and the Fourier neural operator (FNO) (Z. Li et al. 2021). The DeepONet leverages the universal approximation theorem for operators, while the FNO expresses itself as an integral operator with a parameterized Green's function in the Fourier space. Other neural operators, such as the Wavelet neural operator (Tripura et al. 2023) and the Laplace neural operator (Cao et al.

2023), belong to the same class as the FNO, but they are parameterized in the Wavelet and Laplace space, respectively. In our research, we have focused on investigating the DeepONet, but considering the popularity of the FNO, both methods will be addressed theoretically.

The concept of approximating operators marks a paradigm shift from traditional machine learning techniques, which predominantly concentrate on function approximation, to directly solving PDEs. This approach represents a promising advancement in the field.

4.3.1 Neural operators

Let $\Omega \subset \mathbb{R}^D$ be a bounded open set and $\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^{d_x})$ and $\mathcal{Y} = \mathcal{Y}(\Omega; \mathbb{R}^{d_y})$ two separable Banach spaces. Furthermore, assume that $\mathcal{G} : \mathcal{U} \to \mathcal{Y}$ is a non-linear map arising from the solution of a time-dependent PDE. The objective is to approximate the nonlinear operator via the following parametric mapping

$$\mathcal{G} : \mathcal{U} \times \Theta \to \mathcal{Y} \quad \text{or,} \quad \mathcal{G}_\theta : \mathcal{U} \to \mathcal{Y}, \ \theta \in \Theta \tag{4.19}$$

where Θ is a finite-dimensional parameter space. The optimal parameters θ^* are learned via the training of a neural operator with backpropagation based on a dataset $\{\mathbf{u}_j, \boldsymbol{\xi}_j\}_{j=1}^N$ generated on a discretized domain $\Omega_m = \{x_1, \dots, x_m\} \subset \Omega$ where $\{x_j\}_{j=1}^m$ represent the sensor locations, thus $\mathbf{u}_{j|\Omega_m} \in \mathbb{R}^{D_x}$ and $\boldsymbol{\xi}_{j|\Omega_m} \in \mathbb{R}^{D_y}$ where $D_x = d_x \times m$ and $D_y = d_y \times m$.

4.3.2 Fourier neural operators

Fourier neural operators (FNO) introduced by (Z. Li et al. 2021) are based on parameterizing the integral kernel in the Fourier space, and the method can be explained using Green's functions. A solution p to the equation $\mathcal{L}p(x) = f(x)$, where $\mathcal{L}$ is a linear differential operator, can be determined by the superposition of Green's function solutions Gr as

$$p(x) = \int Gr(x, x_0) f(x_0) \mathrm{d}x_0. \tag{4.20}$$

Once Gr is known, all functions p can be found by convolving Gr with f. However, finding Gr is most often difficult or impossible. Therefore, the underlying idea for neural operators is to construct layers that have function inputs as kernel convolutions against the input by replacing the $Gr(x, x_0)$ with a kernel function κ_Θ represented as a neural network. However, Green's functions are only defined for linear operators. To extend the approach to approximating a non-linear operator N to solve $Nu = f(x)$, N is assumed to be locally linear for small enough Δt. A family of T Green's functions $Gr_t(x, x_0)$ represented as $\kappa_t(x, x_0)$ are then stacked together to approximate the non-linear PDE as

$$\int Gr_t(x, x_0) u_t \mathrm{d}x_0 \approx \int \kappa_{\Theta_t}(x, x_0) u_t(x_0) \mathrm{d}x_0, \tag{4.21}$$

where u_0 is the first input lifted from an input $a(x)$ and u_t is the solution from the t'th stacked layer. An iterative solver for iterations $t = 1, \dots, T$ can be formulated in terms of stacked

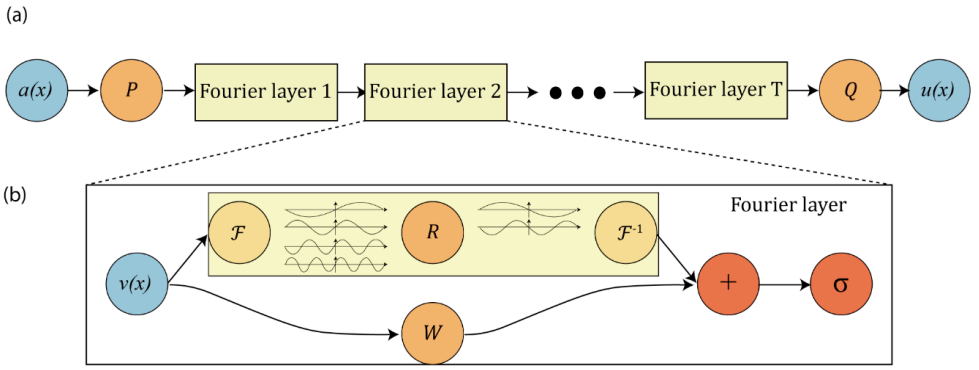


Figure 4.5: Image reproduced from (Z. Li et al. 2021). (a) **The full neural network architecture**: 1) The input function $a(x)$ goes into the network, 2) lift the input to a higher dimension space by the neural network P, 3) pass the function through T stacked Fourier layers, 4) project back the function to the target dimension space by the neural network Q. (b) **The Fourier layers**: 1) the input function v goes into the network, 2) apply the Fourier transform to the input and apply a linear transform R consisting of learnable parameters, apply the inverse Fourier transform, 3) add the output function to the output of a local linear transform W consisting of learnable parameters applied to the input v.

layers as

$$u_{t+1}(x) = \sigma\left(\mathbf{W}_t u_t(x) + \int_D \kappa_{\Theta_t}(x, x_0) u_t(x_0) \mathrm{d}x_0\right), \tag{4.22}$$

where the solution is the result of the last layer $p(x) = u_{T+1}(x)$. The term $\mathbf{W}_t u_t(x)$ has been added to augment the network's capacity to handle non-trivial boundary conditions that might not be possible to capture depending on the kernel used. The network setup proposed in (Z. Li et al. 2021) can be compactly written as

$$p(u) = Q \circ \mathcal{L}_T \circ \cdots \mathcal{L}_2 \circ \mathcal{L}_1 \circ P(u), \tag{4.23}$$

$$\mathcal{L}_t(v)(x) = \sigma\left(W_t v(x) + K_t(v)(x)\right). \tag{4.24}$$

The new terms P and Q are shallow networks, P being the encoder lifting the input to a higher dimension, and Q being the decoder projecting back to the target dimension; $K_t(v)(x) = \int_D \kappa_{\Theta_t}(x, x_0) v(x_0) \mathrm{d}x_0$ is the integral from Equation 4.22. What remains to be handed is how to solve this integral formulation. Many methods have been considered (Kovachki et al. 2023), but the most successful method has been the Fourier neural operators (FNOs) approach (Z. Li et al. 2021). Assuming that the operator is translation invariant, the kernel can be taken to be a convolution kernel, that is

$$\kappa_\theta(x, x_0) = \kappa_\theta(x - x_0). \tag{4.25}$$

The solution to this integral can now be approximated efficiently using fast Fourier methods resulting in multiplication in the Fourier domain instead of convolution in the time domain as

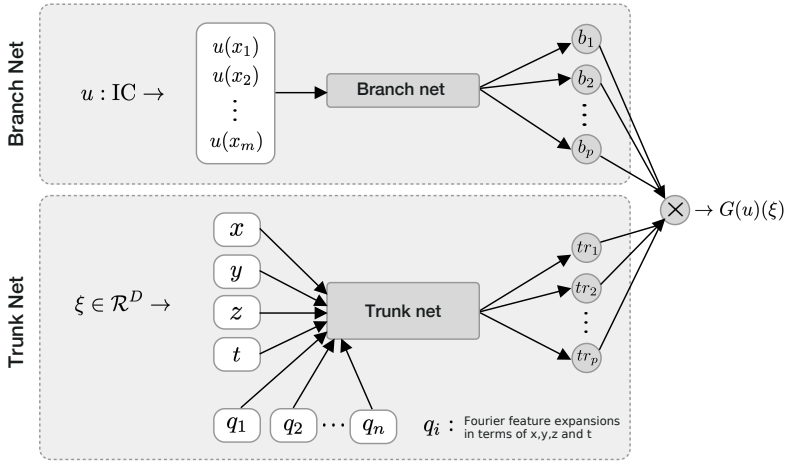
$$K_t(u_t)(x) = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_{\theta_t}) \cdot \mathcal{F}(u_t)\right)(x), \tag{4.26}$$

where $\mathcal{F}$ is the Fourier transform. Now, the task is to learn κ_{θ_t}, but instead $R_{\theta_t} = \mathcal{F}(\kappa_{\theta_t})$ is learned. The FNO architecture is depicted in Figure 4.5.

4.3.3 DeepONet

DeepONet (Lu et al. 2021) is a general deep learning framework for approximating continuous operators contrary to continuous functions, similar to the FNO. The underlying theory stems from the universal operator approximation theorem (Chen et al. 1995), stating that a neural network with a single hidden layer of infinite width can approximate any nonlinear continuous functional or operator. Let G be the operator we want to learn using NNs, defined as $G : u \mapsto G(u)$, where u is the input function to G and $G(u)$ is the output function. For any point, y in the domain of $G(u)$, $G(u)(\xi) \in \mathbb{R}$ is producing a real number. Translating this into a neural network setting, the network takes two inputs, u and ξ, and outputs $G(u)(\xi)$. The input function is discretized by evaluating u at a finite number of points $\{x_i\}$ called 'sensors.' The approximation theorem by Chen & Chen considers shallow networks and only guarantees small approximation errors but does not consider generalization and optimization errors. In (Lu et al. 2021), the authors extended the original theorem by proposing deep

Figure 4.6: DeepONet architecture with parameterized source position for predicting the impulse response for a source/receiver pair over time for a 3D domain. The branch net is taking as input a Gaussian source function **u** determining the source position, sampled at fixed sensor locations. The spatial coordinates x, y, z, and temporal coordinate t are denoted by ξ and are used as input to the trunk net mapping into the output domain of the operator.



neural networks instead of shallow networks and proved that the network is also universal approximators for operators. The proposed deep operator network, DeepONet, achieves small total errors, including approximation, optimization, and generalization errors. The DeepONet architecture depicted in Figure 4.6 consists of two subnetworks, the 'branch net' for the input functions and the 'trunk net' for the locations to evaluate the output function $G(u)$. The trunk network takes ξ as input and outputs $[tr_1, tr_2, \dots, tr_p] \in \mathbb{R}^p$; the branch network takes $[u(x_1), u(x_2), \dots, u(x_m)]^T$ at fixed sensors $\{x_1, x_2, \dots, x_m\}$ and outputs a scalar $b_k \in \mathbb{R}$ for $k = 1, 2, \dots, p$. By merging the trunk and branch in terms of their inner product, we get

$$G(u)(\xi) \approx \sum_{k=1}^{p} \underbrace{b_k \left(u(x_1), u(x_2), \dots, u(x_m)\right)}_{\text{branch}} \underbrace{tr_k(\xi)}_{\text{trunk}} + b_0, \tag{4.27}$$

where b_0 is a trainable bias added for more expressiveness.

The formulation of the Generalized Universal Approximation Theorem for Operators can be found in (Lu et al. 2021) as well as in the Supplementary Material for Paper B. Similar to the function approximation theorem, it only guarantees a small *approximation* errors for a sufficiently large network but does not consider *optimization* or *generalization* errors, and therefore, similar care should be taken as described in subsection 4.2.1 for ensuring good performance.

4.3.4 A comparison between FNO and DeepONet

A debate concerning the performance between the FNO and DeepONet has recently evolved in certain academic circles (Kovachki et al. 2023; Lu, Meng, et al. 2022; Hoop et al. 2022). The intention of this section is not to add to this debate but to summarize the similarities and differences between the methods.

For the FNO, restrictions are imposed on the neural network approximating $Gr(x, x_0)$ so that kernel convolution can be done in the Fourier domain in Equation 4.26. The function u_t is the output from the previous stacked Fourier networks. In DeepONet, the integral is solved directly by calculating the inner product of *two* neural networks, where the trunk net approximates the set of basis functions, and the branch net approximates the corresponding basis function coefficients

$$\text{FNO:} \quad \tilde{p}_t(x) = \sigma\left(\mathbf{W}u_t(x) + \mathcal{F}^{-1}\left(R_{\theta_t} \cdot \mathcal{F}(u_t)\right)(x)\right) \tag{4.28}$$

$$\text{DeepONet:} \quad \tilde{p}(u, \xi) = \sum_{k=1}^{p} \underbrace{b_k \left(u(x_1), u(x_2), \dots, u(x_m)\right)}_{\text{coefficients}} \underbrace{tr_k(\xi)}_{\text{basis functions}} + b_0, \tag{4.29}$$

We have neglected parts of the FNO for clarity, i.e., how the uplifting and stacking are done. Where the role of the Green's function is clear in FNO, it is less clear in DeepONet.

The input data structure to the FNO and DeepONet differs: for the DeepONet, the input function u is separated from the output locations ξ, which can be seen as an inductive bias

$P(\xi|u)$; the input to the FNO is $\{u(x_1), u(x_2)\}$, which allows the network to intrinsically capture the non-linearities (assuming enough Fourier layers). The standard DeepONet structure is a linear approximation of the target operator, where the branch and the trunk net learn the coefficients and the basis, respectively. When the DeepONet can still successfully learn non-linear operators, it is due to the non-linear activation functions in the branch and trunk networks yielding enough expressivity.

A modification to the DeepONet architecture replacing the trunk net with pre-computed POD has been proposed in (Lu, Meng, et al. 2022), bringing them closer to the FNO formulation. The standard FNO formulation assumes the input function to be specified on a periodic domain on a uniform grid and usually works out-of-the-box for these problems (Kovachki et al. 2023). But, as a trade-off, it requires substantial additional treatments to perform well on non-uniform geometries (Lu, Meng, et al. 2022).

4.3.5 Transfer learning

Transfer learning is a technique that leverages knowledge gained from one task (source) to improve the performance of different but related tasks (target). The typical procedure is first to train a source model from scratch on a particular task. Then, to learn a new but related task, the learned model is used as a starting point and adapted to the new task on an eventually smaller dataset. The key idea behind transfer learning is that the knowledge gained in the pre-training on a rich dataset contains valuable information about general patterns and features in the dataset. This knowledge can be used as a foundation and fine-tuned on a new target that differs from the (original) source. This addresses the need for expensive data acquisition and labeling as well as long and expensive training times.

In the context of DeepONet, (Goswami, Kontolati, et al. 2022) have proposed a new transfer learning framework under conditional shift, where the marginal distribution of the source and target data remains the same while the conditional distribution of the output differs; that is $P(\mathbf{x}_s) = P(\mathbf{x}_t)$ and $P(\mathbf{y}_s|\mathbf{x}_s) \neq P(\mathbf{y}_t|\mathbf{x}_t)$, where $\mathbf{x}_s$ and $\mathbf{x}_t$ are the source and target input data, respectively, and $\mathbf{y}_s$ and $\mathbf{y}_t$ are the source and target output data, respectively. The idea behind this method is to train the source model with sufficient data under a standard regression loss and transfer the model to a target model with limited data trained using a hybrid loss consisting of the regression loss and a so-called *conditional embedding operator discrepency* (CEOD) loss. The CEOD loss is used to measure the divergence between conditional distributions. When fine-tuning the target model, the first layers are frozen, and only the parameters for the latter are trained. The CEOD loss was tried for the transfer learning part in Paper B but did not improve the learning compared to solely applying the regression loss for our problem.

Another case where transfer learning can be used is for learning the model to extrapolate. Extrapolating is a notoriously difficult task in machine learning (Barnard et al. 1992; Xu et al. 2021; P. Jin et al. 2020), and transfer learning can be elevated to fit in this context by fine-tuning the model in the extrapolation region. A recent paper by (Zhu et al. 2023) has explored reliable extrapolation for DeepONet via multi-fidelity learning on sparse new observations.

4.3.6 Network architectures (Paper B+C)

Choosing a suitable neural network architecture is important for reducing bias errors (i.e., underfitting). In Paper C, a sensitivity analysis showed that the modified MLP (mod-MLP) outperformed the conventional MLP architecture. Moreover, a comparison between mod-MLP and convolutional neural networks showed that the two architectures were on par. Based on the findings, a similar setup using the mod-MLP was done in Paper B. In the following, we will elaborate on the network architectures.

4.3.6.1 The modified multilayer perceptron

The key extension is the introduction of two encoder networks encoding the input variables to a higher-dimensional feature space. The networks consisting of a single layer are shared between all layers, and a pointwise multiplication operation is performed to update the hidden layers. Let the two encoder networks be denoted $u(\mathbf{x})$ and $v(\mathbf{x})$ and defined as a simple perceptron

$$\begin{aligned} u(\mathbf{x}) &= \sigma(\mathbf{W}_u \mathbf{x} + \mathbf{b}_u), \\ v(\mathbf{x}) &= \sigma(\mathbf{W}_v \mathbf{x} + \mathbf{b}_v), \end{aligned} \tag{4.30}$$

then the mod-MLP is defined as

$$\begin{aligned} \mathbf{y} = ((1 - f_0) \odot u + f_0 \odot v) \circ \\ ((1 - f_1) \odot u + f_1 \odot v) \circ \\ \vdots \\ ((1 - f_n) \odot u + f_n \odot v)(\mathbf{x}), \end{aligned} \tag{4.31}$$

where $\odot$ denotes elementwise multiplication, and $\mathbf{W}_{\{u,v\}}$ and $\mathbf{b}_{\{u,v\}}$ are the weights and biases for the two encoder networks. The architecture is depicted in Figure 4.7, where the encoder networks are applied separately for the branch and trunk net. This differs from the implementation in (S. Wang, H. Wang, et al. 2022), where only two encoder networks are shared between the branch and trunk layer. The motivation behind the mod-MLP is to better propagate information stably through the network since the trainability of the DeepONet depends on merging the branch and trunk net in terms of their inner product only in the last layer. Hence, if the input signals are not properly propagated through the network, this may lead to ineffective training and poor model performance.

4.3.6.2 Convolutional Neural Network

Convolutional neural networks (CNN) are special networks for processing data with a grid-like topology introduced by (LeCun et al. 1998) and use convolution instead of matrix multiplication in one or more layers. In traditional MLPs, each output unit interacts with each input unit through weight parameters describing the interaction. In contrast, CNNs typically have sparse interactions by applying a convolution kernel (much) smaller than the input dimension. Moreover, the convolutional kernel is used for every input position, meaning the

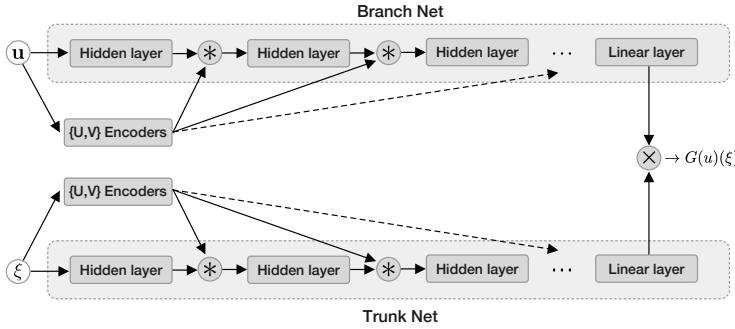
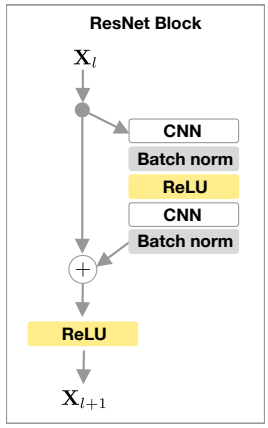


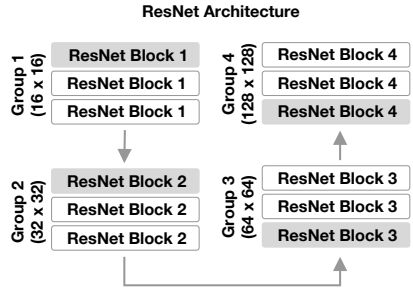
Figure 4.7: The modified MLP architecture applied for both the branch and the trunk net. Two encoders u and v implemented as single-layer neural networks are applied for each MLP, embedding the inputs into a latent space with the size of the layer width of the MLP. The embedded features are then inserted into each hidden layer illustrated by '$*$' performing the operation $(1 - f_i) \odot u + f_i \odot v$.

parameters are shared. Aside from reducing the storage requirement, it also causes the layer to have equivalence to translation.

Although the networks in our work are not particularly deep, we will use the ResNet architecture (He et al. 2016) depicted in Figure 4.8. Several ResNet blocks assemble the ResNet. A ResNet block (Figure 4.8(a)) consists of two stacked CNNs with skip connections and



(a) A ResNet block consisting of two CNNs with batch normalization and `relu` activation functions.



(b) A ResNet architecture consisting of four groups with three ResNet blocks in each group.

Figure 4.8: ResNet architecture consisting of several ResNet blocks.

batch normalization; one or more ResNet blocks comprise a group (all with the same output shape), and one or more groups connect the final ResNet (Figure 4.8(b)). Downsampling takes place in the first block of each group (shown as a gray box) by increasing the strides, and the output channel dimension is increased, forcing the CNN to capture essential features in separate channels. We have used the notation ResNet-{gr1,gr2,gr3,...}, where the element counts inside the square brackets denote the number of groups, and the values denote the number of blocks inside the group indexed by its position. The hidden channel layers are denoted {ch1,ch2,ch3,ch4,...}, where each element index corresponds to the ResNet group with the same index. In Paper C, we showed that using a ResNet-{3,3,3,3} with hidden channel layers {16,32,64,128} for the branch net for approximating the wave equation operator in 2D was on par with using the mod-MLP network.

4.4 Contributions

In Paper A and Paper B, we examine if scientific machine learning models can be used to accurately and efficiently learn the acoustic sound field in dynamic scenes with parametric moving sources incorporating impedance boundaries. In Paper C, we examine how the hyperparameters, choices of architectures, and data resolutions impact the performance of DeepONets in an acoustic setting. The contributions contained in Paper A are:

- A physics-informed neural network method in one dimension is proposed, which learns a compact and efficient surrogate model with parameterized moving Gaussian sources and impedance boundaries and satisfies a system of coupled equations. A data-free approach was taken where only the underlying physics is included in the training and their residual minimized through the loss function.

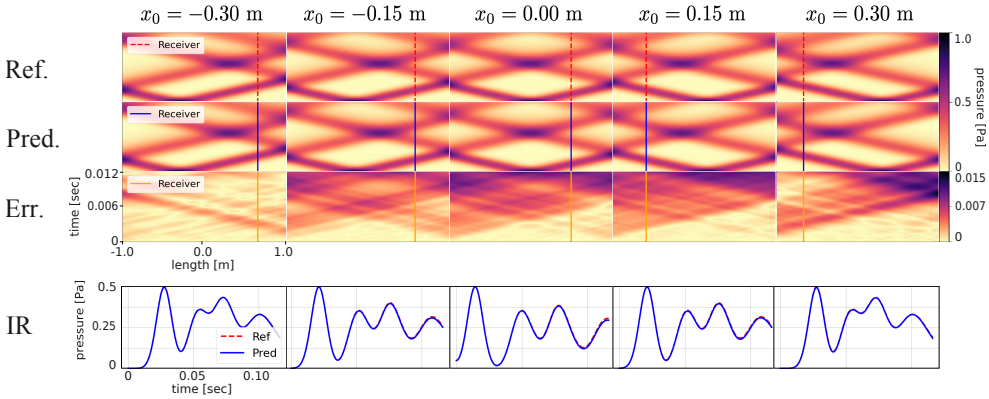


Figure 4.9: Wave propagations in a 1D domain $[-1, 1]$ m for five source positions with frequency-dependent impedance boundaries. Each column corresponds to a source position showing the reference, prediction, error, and impulse response.

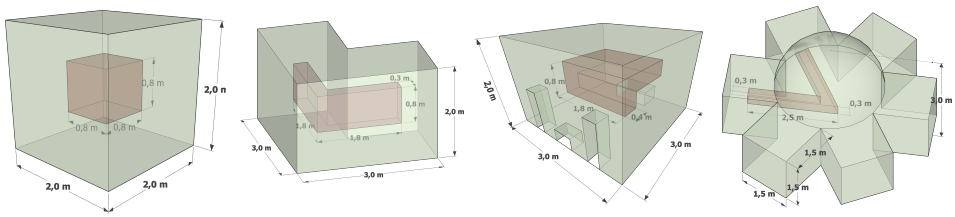


Figure 4.10: Pictorial representations of the four domain geometries adopted in the DeepONet. All the experiments have parametric moving sources allowed to move freely inside a sub-domain of the room shown in shaded red. (a) Cubic room 2 mx2 mx2 m, (b) L-shape room 3 mx3 mx2 m, (c) Furnished room 3 mx3 mx2 m, (d) Dome 36 m^3.

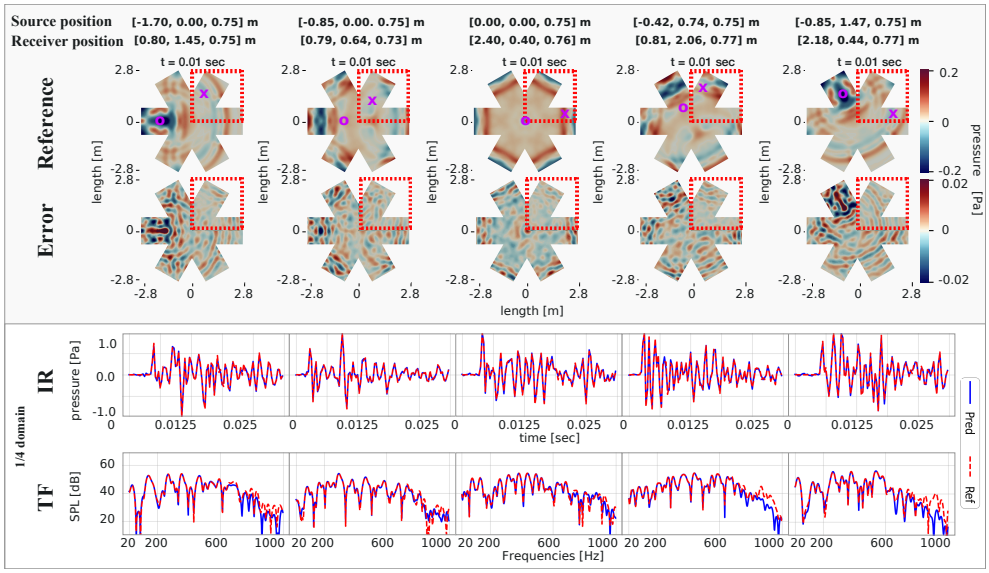


Figure 4.11: Dome 36 m^3. Results show the sound field at $t = 0.01$ s for five parameterized moving sources. The IRs and TFs references and predictions are at the bottom rows for the quarter partition.

- The proposed method alleviates the need for pre-computing the impulse responses and can predict impulse responses for any source/receiver pairs in a grid-less domain in real time. A comparison with a reference solution is conducted for frequency-independent and dependent boundary conditions showing relative mean errors below 2%/0.2 dB. The results for frequency-dependent boundary conditions are depicted in Figure 4.9.

- Two multi-layer feed-forward neural networks have been set up; one predicts the pressure field by solving the PDE, and one solves a system of ODEs predicting the so-called *accumulators* used to calculate the normal velocity. The velocity is, in turn, used to satisfy a boundary condition term solve in terms of pressure.

The contributions contained in Paper B are:

- A DeepONet in three dimensions is proposed, which learns a compact and efficient surrogate model with parameterized moving Gaussian sources and impedance boundaries. The model aims to learn the linear wave-equation operators between infinite-dimensional spaces. In doing so, the model should gain better generalization properties compared to models approximating the function space.

- The proposed method alleviates the need for pre-computing the impulse responses and can predict impulse responses for any source/receiver pairs in a grid-less domain in the millisecond-scale. Our experiments depicted in Figure 4.10 show good agreement with reference solutions, with root mean squared errors ranging from 0.02 Pa to 0.10 Pa. The results for the dome geometry are shown in Figure 4.11.

- To investigate the scalability of the DeepONet, a domain decomposition approach was proposed where separate DeepONets are trained on individual partitions for improved accuracy. This technique showed a much-improved accuracy proposing a way to scale the method to large-scale problems.

- To mitigate the bottleneck of large datasets and expensive training for achieving good predictive performance, we have introduced a simple transfer-learning framework to transfer knowledge between relevant domains explored for 2D domains. A speedup of $3\times$ using 60% of the samples is reported on geometries *close enough*.

- Our method signifies a paradigm shift as no prior machine learning approach has achieved precise predictions of complete wave fields within realistic domains.

The contributions contained in Paper C are:

- A sensitivity analysis is conducted for DeepONets in the context of sound field predictions in 2D. This study aims to understand the intrinsic subtleties of setting up a performant DeepONet.

- The most prominent features impacting performance were the use of `sine` activation functions in combination with the modified MLP. Resolutions of $4-6$ `ppw` for the spatiotemporal data and a source position density of one-fifth of a wavelength are required

for acceptable generalizing properties. The Gaussian input function can be sampled at the Nyquist limit with no performance degradation.

- Suggesting lower bounds on network complexities and data resolutions is especially important to configure tractable DeepONets for 3D geometries. The finding is this study can be used as a starting point, and specifically, the data resolution suggestions can be used directly.

CHAPTER 5

Conclusions and directions for future research

This PhD study investigates methods to efficiently and accurately compute the acoustic sound field in dynamic virtual environments with applications in virtual and augmented reality, computer games, metaverses, and spatial computing. Specifically, the focus has been on calculating sound fields when sources are allowed to move freely, which is notoriously challenging in current numerical methods. Numerical methods are computationally intensive and cannot calculate the sound field in real-time, hence requiring the calculation of the impulse responses for all source/receiver pairs to be done offline. This quickly gets intractable from a storage perspective for large scenes. This thesis aimed to investigate methods that can be used in a real-time setting, not requiring the pre-calculation of impulse responses. During the study, two scientific machine-learning methods have been proposed. The first method uses physics-informed neural networks to learn a surrogate model (Paper A), and the main conclusions are the following:

- Physics-informed neural networks can accurately solve a system of coupled equations for a 1D wave propagation problem with frequency-independent and dependent impedance boundaries and moving sources with relative mean errors below 2%/0.2 dB. Real-time prediction is achieved. No data is needed when including proper knowledge of the physics in terms of the acoustic wave equation.

The second method uses the DeepONet framework (Paper B) to approximate the wave equation operators mapping from one infinity-dimension space to another infinity-dimension space. The knowledge for setting up a performant architecture was gained from the investigation in Paper C for a 2D domain and was only slightly modified for the 3D domain. The main conclusions are the following:

- DeepONet can effectively and accurately learn wave propagations in realistic and complex 3D geometries with frequency-independent and dependent impedance boundary conditions and moving sources. Four geometries are investigated in increasing order of complexity, all showing good accordance with a reference model.
- Applying a domain decomposition technique shows a much-improved accuracy and proposes a way to scale DeepONet to large-scale simulations.
- The model executes very efficiently, achieving millisecond-scale computations. All models executed well below 100 ms, which is considered real-time, except for the dome geometry. The reason is the increased dimensionality of the sensor locations to the branch net, causing a large number of parameters in the input layers. A more

efficient sampling of the sensor location or applying the ResNet architecture proposed in Paper C could decrease the network size and improve the execution time.

Since data is required to efficiently train deep neural networks, efficient methods for generating accurate training data are investigated (Paper D). A domain decomposition method was proposed, where a highly efficient but less flexible Fourier method is applied in rectangular partitions containing air, and the flexible but less effective spectral element method is applied in partitions near the boundaries. An interface handling scheme is proposed to transfer the pressure sound field between the partition. The main conclusions are the following:

- The Fourier-SEM domain decomposition method in 1D indicates a notable performance gain compared to applying the SEM in the full domain. Interface errors are close to -36dB, and the relative mean error is between $5-10\%$, which could be tolerable for interactive virtual reality.
- A major drawback of the method is the need for adding an intermediate SEM layer of 1st order polynomials between the interface and the SEM partition. Due to the non-physical behavior caused by forcing a Neumann boundary condition abruptly at the interface at each time step, shocks are introduced, causing large errors in the approximation to the Laplacian when using high-order polynomials as basis functions.

Future research

Potential directions for future research are listed in the following. First, regarding the scientific machine learning models:

- Training neural operators requires a large amount of data. Paper B showed that 73% of the training time for the 3D DeepONet was spent on data loading. Investigating Physics-Informed DeepONet (S. Wang et al. 2021) should reduce the amount of data needed but add some extra cost for calculating derivates of the differential equations when minimizing their residual for the loss.
- Obtaining large amounts of high-fidelity data takes a lot of computational and storage resources. Multi-fidelity learning (Lu, Pestourie, et al. 2022; Howard et al. 2022; De et al. 2022) could be exploited by learning the underlying features of the dataset by training on low-fidelity data and fine-tuning on a smaller high-fidelity training data.
- For the applications of interest, directional sources are required, which is not considered in this study. However, naively training the DeepONet on a multitude of source directivity patterns quickly gets intractable. Hence, methods for convolving the impulse response obtained from omnidirectional sources with filters obtaining directivities should be investigated.
- Deep neural networks are notoriously challenged when extrapolating outside the training regime. A recent paper by (Zhu et al. 2023) has explored reliable extrapolation for DeepONet via multi-fidelity learning on sparse new observations, and it could be

interesting to apply this method to extrapolate in time or extend geometries for wave propagation problems.

- Finally, a comparison between PINNs and PI-DeepONet concerning training time, prediction accuracy, and evaluation time for 3D wave problems could be informative.

Second, regarding the Fourier-SEM domain decomposition method:

- A remedy to overcome the need to inject a layer of 1st-order polynomials in SEM is most important for developing a scalable method. Further directions in filtering methods should be investigated. So-called *sponge layers* forcing the SEM to satisfy the Neumann boundaries could be considered.
- To properly investigate the Fourier-SEM's numerical stability, accuracy, and performance, a 2D or 3D implementation should be done.
- The Runge-Kutta time stepping scheme should be employed for better accuracy in the SEM partition.

Bibliography

Borrel-Jensen, Nikolas, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Dec. 2021). "Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries." en. In: *Journal of the Acoustical Society of America, Express Letters* 1.12, page 122402.

Borrel-Jensen, Nikolas, Somdatta Goswami, Allan P. Engsig-Karup, George Em Karniadakis, and Cheol-Ho Jeong (2023). "Sound propagation in realistic interactive 3D scenes with parameterized sources using deep neural operators." en. In: *Submitted*.

Borrel-Jensen, Nikolas, Allan P. Engsig-Karup, and Cheol-Ho Jeong (2023). "A sensitivity analysis on the effect of hyperparameters in deep neural operators applied to sound propagation." en. In: *10th Convention of the European Acoustic Association of Forum Acusticum 2023*.

Borrel-Jensen, Nikolas, Allan P. Engsig-Karup, Maarten Hornikx, and Cheol-Ho Jeong (Aug. 2022). "Accelerated sound propagation simulations using an error-free Fourier method coupled with the spectral element method." In: *Proceeding of Inter.noise 2022, Glasgow, Scotland*.

Borrel-Jensen, Nikolas (2023a). *Accelerated sound propagation simulations using an error-free Fourier method coupled with the spectral element method*. Technical report. Technical University of Denmark.

Borrel-Jensen, Nikolas (2023b). *A summary of wave-based boundary element method (WBBEM) with convergence studies*. Technical report. Technical University of Denmark.

Borrel-Jensen, Nikolas, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Oct. 2022a). "DeepONet: Learning the sound propagation in 1D with parameterized sources using deep learning for approximating the wave equation operators." en. In: *Proceeding of the 24th International Congress on Acoustics*.

Borrel-Jensen, Nikolas, Allan P. Engsig-Karup, and Cheol-Ho Jeong (Apr. 2022b). "Machine learning-based room acoustics using flow maps and physics-informed neural networks." In: *Presented at the 182nd Meeting of the Acoustical Society of America, 23-26 May, Denver* 151, A232–A233. ISSN: 0001-4966.

Melander, Anders et al. (2020). "Massive parallel nodal discontinuous Galerkin finite element method simulator for room acoustics." In: *International Journal of High Performance Computing Applications*. URL: http://infoscience.epfl.ch/record/279868.

Plack, Christopher J. (2018). *The Sense of Hearing*. 3rd. Taylor and Francis. ISBN: 9781351802765.

Meta (2020). *The Future of Audio*. URL: https://about.fb.com/news/2020/09/facebook-reality-labs-research-future-of-audio/ (visited on 06/22/2023).

Meta (2021). *Introducing Ray-Ban Stories: First-Generation Smart Glasses (press release)*. URL: https://about.fb.com/news/2021/09/introducing-ray-ban-stories-smart-glasses/ (visited on 06/22/2023).

Apple (2022). *Apple announces the next generation of AirPods Pro*. URL: https://www.apple.com/newsroom/2022/09/apple-announces-the-next-generation-of-airpods-pro/ (visited on 06/22/2023).

Greenwold, Simon (2003). "Spatial Computing." Master's thesis. Massachusetts Institute of Technology.

Meta (2022). *Meta Quest Pro Is Now Available*. URL: https://about.fb.com/news/2022/10/meta-quest-pro-is-now-available/ (visited on 06/22/2023).

Apple (2023). *Introducing Apple Vision Pro: Apples first spatial computer (press release)*. URL: https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/ (visited on 06/22/2023).

Savioja, Lauri and Peter U. Svensson (Aug. 2015). "Overview of geometrical room acoustic modeling techniques." en. In: *Journal of the Acoustical Society of America* 138.2, pages 708–730. ISSN: 0001-4966. DOI: https://doi.org/10.1121/1.4926438.

Sandvad, J. (May 1996). "Dynamic Aspects of Auditory Virtual Environments." In: *Audio Engineering Society Convention 100*.

Wefers, Frank and Michael Vorländer (Nov. 2018). "Flexible data structures for dynamic virtual auditory scenes." en. In: *Virtual Reality* 22.4, pages 281–295. ISSN: 1359-4338, 1434-9957. DOI: https://doi.org/10.1007/s10055-018-0332-9.

Pelzer, Sönke, Lukas Aspöck, Dirk Schröder, and Michael Vorländer (Mar. 2014). "Interactive Real-Time Simulation and Auralization for Modifiable Rooms." en. In: *Building Acoustics* 21.1, pages 65–73. ISSN: 1351-010X, 2059-8025. DOI: https://doi.org/10.1260/1351-010X.21.1.65.

Lauterbach, Christian, Anish Chandak, and Dinesh Manocha (Nov. 2007). "Interactive sound rendering in complex and dynamic scenes using frustum tracing." en. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6, pages 1672–1679. ISSN: 1077-2626. DOI: https://doi.org/10.1109/TVCG.2007.70567.

Schröder, Dirk (Jan. 2011). "Physically Based Real-Time Auralization of Interactive Virtual Environments." PhD thesis.

Raven (2023). URL: https://www.virtualacoustics.org/RAVEN (visited on 07/29/2023).

spatializer, Oculus (2023). URL: https://developer.oculus.com/downloads/package/oculus-spatializer-unity/ (visited on 07/29/2023).

Audio, Steam (2023). URL: https://valvesoftware.github.io/steam-audio/ (visited on 07/29/2023).

Resonance, Google (2023). URL: https://resonance-audio.github.io/resonance-audio/ (visited on 07/29/2023).

Wwise, Audiokinetic (2023). URL: https://www.audiokinetic.com/products/wwise/ (visited on 07/29/2023).

Fmod (2023). URL: https://www.fmod.com/ (visited on 07/29/2023).

Okuzono, Takeshi, Toru Otsuru, Reiji Tomiku, and Noriko Okamoto (2014). "A finite-element method using dispersion reduced spline elements for room acoustics simulation." In: *Applied Acoustics* 79, pages 1–8. ISSN: 0003682X.

Pind, Finnur et al. (2019). "Time domain room acoustic simulations using the spectral element method." In: *Journal of the Acoustical Society of America* 145.6, pages 3299–3310. ISSN: 0001-4966.

Botteldoorena, Dick (1995). "Finite-difference time-domain simulation of low-frequency room acoustic problems." In: *Journal of the Acoustical Society of America* 98.6, pages 3302–3308.

Hamilton, Brian and Stefan Bilbao (2017). "FDTD Methods for 3-D Room Acoustics Simulation with High-Order Accuracy in Space and Time." In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 25.11, pages 2112–2124. ISSN: 23299290.

Microsoft (2023). URL: `https://learn.microsoft.com/en-us/gaming/acoustics/what-is-acoustics` (visited on 07/29/2023).

Unity (2023). URL: `https://unity.com` (visited on 06/22/2023).

Unreal (2023). URL: `https://www.unrealengine.com` (visited on 06/22/2023).

Treble (2023). URL: `https://www.treble.tech` (visited on 06/22/2023).

Turner, Daniel, Damian Murphy, Chris Pike, and Chris Baume (2022). "Spatial audio production for immersive media experiences: Perspectives on practice-led approaches to designing immersive audio content." In: *Soundtrack, The* 13.1, pages 73–94. ISSN: 1751-4207. DOI: `https://doi.org/10.1386/ts_00017_1`.

Brinkmann, Fabian et al. (Apr. 2019). "A round robin on room acoustical simulation and auralization." en. In: *The Journal of the Acoustical Society of America* 145.4, pages 2746–2760. ISSN: 0001-4966, 1520-8524. DOI: `https://doi.org/10.1121/1.5096178`.

Pind, Finnur (2020). "Wave-Based Virtual Acoustics." English. PhD thesis.

Wightman, Frederic L. and Doris J. Kistler (Mar. 1992). "The dominant role of lowfrequency interaural time differences in sound localization." In: *The Journal of the Acoustical Society of America* 91.3, pages 1648–1661. ISSN: 0001-4966. DOI: `https://doi.org/10.1121/1.402445`.

Raghuvanshi, Nikunj and John Snyder (2014). "Parametric wave field coding for precomputed sound propagation." In: *ACM Transactions on Graphics* 33.4. ISSN: 15577333.

Raghuvanshi, Nikunj (2021). *Dynamic Portal Occlusion for Precomputed Interactive Sound Propagation*. arXiv: `2107.11548`. URL: `arxiv.org/abs/2107.11548`.

Bianco, Michael J. et al. (Nov. 2019). "Machine learning in acoustics: theory and applications." en. In: *Journal of the Acoustical Society of America* 146.5, pages 3590–3628. ISSN: 0001-4966. DOI: `https://doi.org/10.1121/1.5133944`.

Cai, Shengze, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis (Dec. 2021). "Physics-informed neural networks (PINNs) for fluid mechanics: a review." en. In: *Acta Mechanica Sinica* 37.12, pages 1727–1738. ISSN: 0567-7718, 1614-3116. DOI: `https://doi.org/10.1007/s10409-021-01148-1`.

Cuomo, Salvatore et al. (June 2022). *Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next*. en. URL: `http://arxiv.org/abs/2201.05624`.

Psichogios, Dimitris C. and Lyle H. Ungar (1992). "A hybrid neural networkfirst principles approach to process modeling." In: *AIChE Journal* 38.10, pages 1499–1511. DOI: `https://doi.org/10.1002/aic.690381003`.

Lagaris, I.E., A. Likas, and D.I. Fotiadis (1998). "Artificial neural networks for solving ordinary and partial differential equations." In: *IEEE Transactions on Neural Networks* 9.5, pages 987–1000.

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." In: *Journal of Computational Physics* 378, pages 686–707. ISSN: 10902716.

Moseley, B., A. Markham, and T. Nissen-Meyer (2020). *Solving the wave equation with physics-informed deep learning*. arXiv: 2006.11894. URL: arxiv.org/abs/2006.11894.

Rasht-Behesht, Majid, Christian Huber, Khemraj Shukla, and George Em Karniadakis (2021). *Physics-informed Neural Networks (PINNs) for Wave Propagation and Full Waveform Inversions*. URL: arxiv.org/abs/2108.12035.

Ma, Fei, Thushara D. Abhayapala, Prasanga N. Samarasinghe, and Xingyu Chen (July 2023). *Physics Informed Neural Network for Head-Related Transfer Function Upsampling*. en. URL: http://arxiv.org/abs/2307.14650.

Alkhadhr, Shaikhah, Xilun Liu, and Mohamed Almekkawy (Sept. 2021). "Modeling of the Forward Wave Propagation Using Physics-Informed Neural Networks." en. In: *2021 IEEE International Ultrasonics Symposium (IUS)*. Xi'an, China: IEEE, pages 1–4. ISBN: 978-1-66540-355-9. DOI: https://doi.org/10.1109/IUS52206.2021.9593574.

Lee, Soo Young, Choon-Su Park, Keonhyeok Park, Hyung Jin Lee, and Seungchul Lee (Aug. 2023). "A Physics-informed and data-driven deep learning approach for wave propagation and its scattering characteristics." en. In: *Engineering with Computers* 39.4, pages 2609–2625. ISSN: 0177-0667, 1435-5663. DOI: https://doi.org/10.1007/s00366-022-01640-7.

Li, Zongyi et al. (May 2021). *Fourier Neural Operator for Parametric Partial Differential Equations*. en. URL: http://arxiv.org/abs/2010.08895.

Lu, Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis (2021). "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators." In: *Nature Machine Intelligence* 3.3, pages 218–229. ISSN: 25225839. DOI: https://doi.org/10.1038/s42256-021-00302-5.

Cao, Qianying, Somdatta Goswami, and George Em Karniadakis (2023). *LNO: Laplace Neural Operator for Solving Differential Equations*.

Chen, Tianping and Hong Chen (1995). "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems." In: *IEEE Transactions on Neural Networks* 6.4, pages 911–917. DOI: https://doi.org/10.1109/72.392253.

Goswami, Somdatta, Minglang Yin, Yue Yu, and George Em Karniadakis (2022). "A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials." In: *Computer Methods in Applied Mechanics and Engineering* 391, page 114587.

Kumar, Varun, Somdatta Goswami, Daniel J Smith, and George Em Karniadakis (2023). "Real-time prediction of multiple output states in diesel engines using a deep neural operator framework." In.

Oommen, Vivek, Khemraj Shukla, Somdatta Goswami, Rémi Dingreville, and George Em Karniadakis (2022). "Learning two-phase microstructure evolution using neural operators and autoencoder architectures." In: *npj Computational Materials* 8.1, page 190.

Lin, Chensen et al. (2021). "Operator learning for predicting multiscale bubble growth dynamics." In: *Journal of Chemical Physics* 154.10, page 104118.

Goswami, Somdatta, David S. Li, et al. (2022). "Neural operator learning of heterogeneous mechanobiological insults contributing to aortic aneurysms." In: *Journal of the Royal Society Interface* 19.193, page 20220410.

Shukla, Khemraj et al. (2023). "Deep neural operators can serve as accurate surrogates for shape optimization: a case study for airfoils." In: *arXiv preprint arXiv:2302.00807*.

Hesthaven, Jan S., Gianluigi Rozza, and Benjamin Stamm (2015). *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, pages 1–131.

Llopis, Hermes Sampedro, Allan P. Engsig-Karup, Cheol-Ho Jeong, Finnur Pind, and Jan S. Hesthaven (Aug. 2022). "Reduced basis methods for numerical room acoustic simulations with parametrized boundaries." In: *Journal of the Acoustical Society of America* 152 (2), pages 851–865. ISSN: 0001-4966. DOI: https://doi.org/10.1121/10.0012696.

Savioja, Lauri, Tapini Rinne, and Tapio Takala (1994). "Simulation of Room Acoustics with a 3-D Finite Difference Mesh." English. In: *The 1994 International Computer Music Conference, Aarhus, September 12-17, 1994*. United States: International Computer Music Association ICMA, pages 463–466.

Kowalczyk, Konrad and Maarten Van Walstijn (Jan. 2011). "Room Acoustics Simulation Using 3-D Compact Explicit FDTD Schemes." en. In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.1, pages 34–46. ISSN: 1558-7916, 1558-7924. DOI: https://doi.org/10.1109/TASL.2010.2045179.

Borrel-Jensen, Nikolas (2012). "Real-time Auralisation of the Lower Frequency Sound Field Using Numerical Methods on the GPU." University of Copenhagen/RWTH Aachen University.

Van Mourik, Jelle and Damian Murphy (Dec. 2014). "Explicit Higher-Order FDTD Schemes for 3D Room Acoustic Simulation." en. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.12, pages 2003–2011. ISSN: 2329-9290, 2329-9304. DOI: 10.1109/TASLP.2014.2341913. URL: http://ieeexplore.ieee.org/document/6862889/ (visited on 08/04/2023).

Hamilton, Brian (Nov. 2016). "Finite Difference and Finite Volume Methods for Wave-based Modelling of Room Acoustics." PhD thesis. DOI: https://doi.org/10.13140/RG.2.2.31081.70240.

Bilbao, Stefan (July 2013). "Modeling of Complex Geometries and Boundary Conditions in Finite Difference/Finite Volume Time Domain Room Acoustics Simulation." en. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.7, pages 1524–1533. ISSN: 1558-7916, 1558-7924. DOI: https://doi.org/10.1109/TASL.2013.2256897.

Bilbao, Stefan, Brian Hamilton, Jonathan Botts, and Lauri Savioja (Jan. 2016). "Finite Volume Time Domain Room Acoustics Simulation under General Impedance Boundary Conditions." en. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.1, pages 161–173. ISSN: 2329-9290, 2329-9304. DOI: https://doi.org/10.1109/TASLP.2015.2500018.

Chobeau, Pierre, Sebastian Prepelita, Jukka Saarelma, Jonathan Botts, and Lauri Savioja (Jan. 2016). "Finite Volume Time Domain Simulations of Frequency-Dependent Boundary Conditions and Absorbing Layer." In: *Audio Engineering Society Conference: 60th In-*

ternational Conference: DREAMS (Dereverberation and Reverberation of Audio, Music, and Speech).

Webb, Craig J. and Stefan Bilbao (May 2011). "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity." en. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Prague, Czech Republic: IEEE, pages 317–320. ISBN: 978-1-4577-0538-0. DOI: 10.1109/ICASSP.2011.5946404. URL: http://ieeexplore.ieee.org/document/5946404/ (visited on 06/21/2023).

Craggs, A. (1994). "A Finite Element Method for the Free Vibration of Air in Ducts and Rooms with Absorbing Walls." In: *Journal of Sound and Vibration* 173.4, pages 568–576. ISSN: 0022-460X. DOI: https://doi.org/10.1006/jsvi.1994.1553.

Okuzono, Takeshi and Kimihiro Sakagami (2018). "A frequency domain finite element solver for acoustic simulations of 3D rooms with microperforated panel absorbers." In: *Applied Acoustics* 129, pages 1–12. ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2017.07.008.

Hesthaven, Jan S. and Tim Warburton (2008). *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. eng. Volume 54. Springer, XIV, 500 S. ISBN: 0387720650.

Hargreaves, Jonathan A., Luke R. Rendell, and Yiu W. Lam (Apr. 2019). "A framework for auralization of boundary element method simulations including source and receiver directivity." In: *Journal of the Acoustical Society of America* 145.4, pages 2625–2637. ISSN: 0001-4966. DOI: https://doi.org/10.1121/1.5096171. eprint: https://pubs.aip.org/asa/jasa/article-pdf/145/4/2625/16701273/2625_1_online.pdf.

Canuto, Claudio, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang (2006). *Spectral Methods: Fundamentals in Single Domains*. Springer.

Raghuvanshi, Nikunj, Rahul Narain, and Ming C. Lin (Sept. 2009a). "Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition." en. In: *IEEE Transactions on Visualization and Computer Graphics* 15.5, pages 789–801. ISSN: 1077-2626. DOI: https://doi.org/10.1109/TVCG.2009.28.

Muñoz, Raúl Pagán and Maarten Hornikx (Nov. 2017). "Hybrid Fourier pseudospectral/discontinuous Galerkin time-domain method for wave propagation." en. In: *Journal of Computational Physics* 348, pages 416–432. ISSN: 00219991. DOI: https://doi.org/10.1016/j.jcp.2017.07.046.

Desmet, Wim (1998). "A wave based prediction technique for coupled vibro-acoustic analysis." PhD thesis. KU Leuven.

Van Genechten, Bert et al. (Jan. 2012a). "An efficient Wave Based Method for solving Helmholtz problems in three-dimensional bounded domains." en. In: *Engineering Analysis with Boundary Elements* 36.1, pages 63–75. ISSN: 09557997. DOI: https://doi.org/10.1016/j.enganabound.2011.07.011.

Pluymers, Bert (2006). "Wave based modelling methods for steady-state vibro-acoustics." nl. PhD thesis.

Deckers, Elke et al. (June 2014). "The wave based method: An overview of 15 years of research." en. In: *Wave Motion* 51.4, pages 550–565. ISSN: 01652125.

Van Genechten, Bert et al. (2012b). "An efficient Wave Based Method for solving Helmholtz problems in three-dimensional bounded domains." In: *Engineering Analysis with Bound-*

ary Elements 36.1. Special Issue on Trefftz Method, pages 63–75. ISSN: 0955-7997. DOI: https://doi.org/10.1016/j.enganabound.2011.07.011.

Raghuvanshi, Nikunj, Rahul Narain, and Ming C. Lin (Sept. 2009b). "Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition." In: *IEEE Transactions on Visualization and Computer Graphics* 15 (5), pages 789–801.

Morales, Nicolas, Ravish Mehra, and Dinesh Manocha (Oct. 2015). "A parallel time-domain wave simulator based on rectangular decomposition for distributed memory architectures." en. In: *Applied Acoustics* 97, pages 104–114. ISSN: 0003682X. DOI: https://doi.org/10.1016/j.apacoust.2015.03.017.

Dissanayake, M. W. M. G. and N. Phan-Thien (1994). "Neural-network-based approximations for solving partial differential equations." In: *Communications in Numerical Methods in Engineering* 10.3, pages 195–201. DOI: https://doi.org/10.1002/cnm.1640100303.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature* 323.6088, pages 533–536.

Nielsen, Michael A. (2017). *How the backpropagation algorithm works*. URL: http://neuralnetworksanddeeplearning.com/chap2.html.

Lanthaler, Samuel, Siddhartha Mishra, and George E Karniadakis (2022). "Error estimates for deeponets: A deep learning framework in infinite dimensions." In: *Transactions of Mathematics and Its Applications* 6.1, tnac001.

Rahaman, Nasim et al. (June 2018). *On the Spectral Bias of Neural Networks*. URL: http://arxiv.org/abs/1806.08734.

Basri, Ronen et al. (Mar. 2020). "Frequency Bias in Neural Networks for Input of Non-Uniform Density." In: URL: http://arxiv.org/abs/2003.04560.

Benbarka, Nuri, Timon Hofer, Hamd Ul-Moqeet Riaz, and Andreas Zell (Jan. 2022). "Seeing Implicit Neural Representations as Fourier Series." en. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, pages 2283–2292. ISBN: 978-1-66540-915-5. DOI: https://doi.org/10.1109/WACV51458.2022.00234.

Glorot, Xavier and Yoshua Bengio (May 2010). "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Edited by Yee Whye Teh and Mike Titterington. Volume 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pages 249–256.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification." In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, pages 1026–1034.

Sitzmann, Vincent, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein (2020). *Implicit Neural Representations with Periodic Activation Functions*. arXiv: 2006.09661. URL: arxiv.org/abs/2006.09661.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators." In: *Neural Networks* 2.5, pages 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

Troian, Renata, Didier Dragna, Christophe Bailly, and Marie Annick Galland (2017). "Broadband liner impedance eduction for multimodal acoustic propagation in the presence of a mean flow." In: *Journal of Sound and Vibration* 392, pages 200–216. ISSN: 10958568.

Haghighat, Ehsan and Ruben Juanes (2021). "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks." In: *Computer Methods in Applied Mechanics and Engineering* 373, page 113552. ISSN: 00457825. arXiv: 2005.08803. URL: https://www.sciann.com.

Wang, Sifan, Xinling Yu, and Paris Perdikaris (2022). "When and why PINNs fail to train: A neural tangent kernel perspective." In: *Journal of Computational Physics* 449, page 110768. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2021.110768.

Jin, Xiaowei, Shengze Cai, Hui Li, and George Em Karniadakis (2021). "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations." In: *Journal of Computational Physics* 426, page 109951. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.109951.

Bischof, Rafael and Michael Kraus (2021). "Multi-Objective Loss Balancing for Physics-Informed Deep Learning." en. In: URL: https://doi.org/10.48550/arXiv.2110.09813.

McClenny, Levi D. and Ulisses M. Braga-Neto (Feb. 2023). "Self-adaptive physics-informed neural networks." en. In: *Journal of Computational Physics* 474, page 111722. ISSN: 00219991. DOI: https://doi.org/10.1016/j.jcp.2022.111722.

Wu, Chenxi, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu (Jan. 2023). "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks." en. In: *Computer Methods in Applied Mechanics and Engineering* 403, page 115671. ISSN: 00457825. DOI: https://doi.org/10.1016/j.cma.2022.115671.

Wang, Sifan, Shyam Sankaran, and Paris Perdikaris (Mar. 2022). *Respecting causality is all you need for training physics-informed neural networks*. en. URL: http://arxiv.org/abs/2203.07404.

Daw, Arka, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne (June 2023). *Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling*. en. URL: http://arxiv.org/abs/2207.02338.

Tripura, Tapas and Souvik Chakraborty (2023). "Wavelet Neural Operator for solving parametric partial differential equations in computational mechanics problems." In: *Computer Methods in Applied Mechanics and Engineering* 404, page 115783.

Kovachki, Nikola et al. (Apr. 2023). *Neural Operator: Learning Maps Between Function Spaces*. en. arXiv:2108.08481 [cs, math]. URL: http://arxiv.org/abs/2108.08481.

Lu, Lu, Xuhui Meng, et al. (Apr. 2022). "A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data." en. In: *Computer Methods in Applied Mechanics and Engineering* 393, page 114778. ISSN: 00457825. DOI: https://doi.org/10.1016/j.cma.2022.114778.

Hoop, Maarten V. de, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart (Aug. 2022). *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*. en. URL: http://arxiv.org/abs/2203.13181.

Goswami, Somdatta, Katiana Kontolati, Michael D. Shields, and George Em Karniadakis (2022). "Deep transfer operator learning for partial differential equations under conditional shift." In: *Nature Machine Intelligence* 4.12, pages 1155–1164.

Barnard, E. and L.F.A. Wessels (1992). "Extrapolation and interpolation in neural network classifiers." In: *IEEE Control Systems Magazine* 12.5, pages 50–53. DOI: https://doi.org/10.1109/37.158898.

Xu, Keyulu et al. (2021). *How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks*. arXiv: 2009.11848 [cs.LG].

Jin, Pengzhan, Lu Lu, Yifa Tang, and George Em Karniadakis (2020). "Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness." In: *Neural Networks* 130, pages 85–99. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2020.06.024.

Zhu, Min, Handi Zhang, Anran Jiao, George Em Karniadakis, and Lu Lu (July 2023). "Reliable extrapolation of deep neural operators informed by physics or sparse observations." en. In: *Computer Methods in Applied Mechanics and Engineering* 412, page 116064. ISSN: 00457825. DOI: https://doi.org/10.1016/j.cma.2023.116064.

Wang, Sifan, Hanwen Wang, and Paris Perdikaris (Aug. 2022). "Improved Architectures and Training Algorithms for Deep Operator Networks." en. In: *Journal of Scientific Computing* 92.2, page 35. ISSN: 0885-7474, 1573-7691. DOI: https://doi.org/10.1007/s10915-022-01881-0.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pages 2278–2324.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. DOI: https://doi.org/10.1109/CVPR.2016.90.

Wang, Sifan, Hanwen Wang, and Paris Perdikaris (2021). "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets." In: *Science Advances* 7.40. DOI: https://doi.org/10.1126/sciadv.abi8605.

Lu, Lu, Raphaël Pestourie, Steven G. Johnson, and Giuseppe Romano (June 2022). "Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport." en. In: *Physical Review Research* 4.2, page 023210. ISSN: 2643-1564. DOI: https://doi.org/10.1103/PhysRevResearch.4.023210.

Howard, Amanda A., Mauro Perego, George E. Karniadakis, and Panos Stinis (Apr. 2022). *Multifidelity Deep Operator Networks*. en. URL: http://arxiv.org/abs/2204.09157.

De, Subhayan, Matthew Reynolds, Malik Hassanaly, Ryan N. King, and Alireza Doostan (Aug. 2022). *Bi-fidelity Modeling of Uncertain and Partially Unknown Systems using DeepONets*. en. URL: http://arxiv.org/abs/2204.00997.

Publications

Paper A

JASA EXPRESS LETTERS ARTICLE asa.scitation.org/journal/jel

Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries

Nikolas Borrel-Jensen,[1,a] **Allan P. Engsig-Karup,**[2,b] **and Cheol-Ho Jeong**[1,c]
[1]*Acoustic Technology, Department of Electrical Engineering, Technical University of Denmark, 2800 Kongens Lyngby, Denmark*
[2]*Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kongens Lyngby, Denmark*
nibor@elektro.dtu.dk, apek@dtu.dk, chj@elektro.dtu.dk

Abstract: Realistic sound is essential in virtual environments, such as computer games and mixed reality. Efficient and accurate numerical methods for pre-calculating acoustics have been developed over the last decade; however, pre-calculating acoustics makes handling dynamic scenes with moving sources challenging, requiring intractable memory storage. A physics-informed neural network (PINN) method in one dimension is presented, which learns a compact and efficient surrogate model with parameterized moving Gaussian sources and impedance boundaries and satisfies a system of coupled equations. The model shows relative mean errors below 2%/0.2 dB and proposes a first step in developing PINNs for realistic three-dimensional scenes.

[Editor: D. Keith Wilson] https://doi.org/10.1121/10.0009057

Received: 22 September 2021 **Accepted:** 20 November 2021 **Published Online:** 27 December 2021

1. Introduction

In computer games and mixed reality, realistic sound is essential for an immersive user experience. The impulse responses (IR) can be obtained accurately and efficiently by numerically solving the wave equation using traditional numerical methods, such as finite element methods,[1] spectral element methods (SEM),[2] discontinuous Galerkin finite element method,[3] and finite-difference time-domain methods.[4,5] For real-time applications spanning a broad frequency range, the IRs are calculated offline due to the computational requirements. However, for dynamic, interactive scenes with numerous moving sources and receivers, the computation time and storage requirement for a lookup database become intractable (in the range of gigabytes) since the IR is calculated for each source-receiver pair. When covering the whole audible frequency range, these challenges become even more extensive. Previous attempts to overcome the storage requirements of the IRs include work for lossy compression,[6] and lately, a novel portal search method has been proposed as a drop-in solution to pre-computed IRs to adapt to flexible scenes, e.g., when doors and windows are opened and closed.[7] A recent technique for handling parameter parameterization and model order reduction for acceleration of numerical models is the reduced basis method (RBM).[8,9] Although very efficient, RBM cannot meet the runtime requirements regarding computation time for virtual acoustics.

In this paper, we consider a new approach using physics-informed neural networks (PINNs)[10–12] including knowledge of the underlying physics (in contrary to traditional "black box" neural networks[13]) to learn a surrogate model for a one-dimensional (1D) domain that can be executed very efficiently at runtime (in the range of ms) and takes up little storage due to their intrinsic interpolation properties in grid-less domains. The applications of PINNs in virtual acoustics are very limited,[14,15] and the main contribution of this work is the development of frequency-dependent and independent impedance boundary conditions with parameterized moving Gaussian sources, making it possible to model sound propagation taking boundary materials properly into account. This work investigated PINNs for virtual acoustics in a 1D domain—still taking the necessary physics into account—making it a possible stepping stone to model realistic and complex 3D scenes for applications, such as games and mixed reality, where the computation and storage requirements are very strict.

[a]Author to whom correspondence should be addressed, ORCID: 0000-0002-8820-4635.
[b]ORCID: 0000-0001-8626-1575.
[c]ORCID: 0000-0002-9864-7317.

2. Methods

We take a data-free approach where only the underlying physics is included in the training and their residual minimized through the loss function, allowing insights into how well PINNs perform for predicting sound fields in acoustic conditions. The Gaussian impulse is used as the initial condition tested with frequency-independent and dependent impedance boundaries. To assess the quality of the developed PINN models, we have used our in-house open-source SEM simulators.[2]

2.1 Governing equations

We consider in the following the use of PINNs for the construction of a surrogate model predicting the solution to the linear wave equation in 1D,

$$\frac{\partial^2 p(x,t)}{\partial t^2} - c^2 \frac{\partial^2 p(x,t)}{\partial x^2} = 0, \quad t \in \mathbb{R}^+, \quad x \in \mathbb{R}, \tag{1}$$

where p is the pressure (Pa), t is the time (s), and c is the speed of sound in air (m/s). The initial conditions (ICs) are satisfied by using a Gaussian source for the pressure part and setting the velocity equal to zero,

$$p(x, t=0, x_0) = \exp\left[-\left(\frac{x-x_0}{\sigma_0}\right)^2\right], \quad \frac{\partial p(x, t=0, x_0)}{\partial t} = 0, \tag{2}$$

with σ_0 being the width of the pulse determining the frequencies to span.

2.2 Boundary conditions

We consider impedance boundaries and denote the boundary domain as Γ (in 1D, the left and right endpoints). We will omit the source position x_0 in the following.

2.2.1 Frequency-independent impedance boundaries

The acoustic properties of a wall can be described by its surface impedance[16] $Z_s = p/v_n$, where v_n is the normal component of the velocity at the same location on the wall surface. Combining the surface impedance with the pressure term $\partial p/\partial \mathbf{n} = -\rho_0(\partial v_n/\partial t)$ of the linear coupled wave equation yields

$$\frac{\partial p}{\partial t} = -c\xi \frac{\partial p}{\partial \mathbf{n}}, \tag{3}$$

where $\xi = Z_s/(\rho_0 c)$ is the normalized surface impedance and ρ_0 denotes the air density (kg/m^3). Note that perfectly reflecting boundaries can be obtained by letting $\xi \to \infty$ being the Neumann boundary formulation.

2.2.2 Frequency-dependent impedance boundaries

The wall impedance can be written as a rational function in terms of the admittance $Y = 1/Z_s$ and rewritten by using partial fraction decomposition in the last equation (17)

$$Y(\omega) = \frac{a_0 + \cdots + a_N(-i\omega)^N}{1 + \cdots + b_N(-i\omega)^N} = Y_\infty + \sum_{k=0}^{Q-1} \frac{A_k}{\lambda_k - i\omega} + \sum_{k=0}^{S-1}\left(\frac{B_k + iC_k}{\alpha_k + i\beta_k - i\omega} + \frac{B_k - iC_k}{\alpha_k - i\beta_k - i\omega}\right), \tag{4}$$

where a, b are real coefficients; $i = \sqrt{-1}$ is the complex number; Q is the number of real poles λ_k; S is the number of complex conjugate pole pairs $\alpha_k \pm j\beta_k$; and Y_∞, A_k, B_k, and C_k are numerical coefficients. Since we are concerned with the (time-domain) wave equation, the inverse Fourier transform is applied on the admittance and on the partial fraction decomposition term in Eq. (4). Combining these gives[17]

$$v_n(t) = Y_\infty p(t) + \sum_{k=0}^{Q-1} A_k \phi_k(t) + \sum_{k=0}^{S-1} 2\left[B_k \psi_k^{(0)}(t) + C_k \psi_k^{(1)}(t)\right]. \tag{5}$$

The functions ϕ_k, $\psi_k^{(0)}$, and $\psi_k^{(1)}$ are the so-called accumulators determined by the following set of ordinary differential equations (ODEs) referred to as auxiliary differential equations (ADEs):

$$\frac{d\phi_k}{dt} + \lambda_k \phi_k = p, \quad \frac{d\psi_k^{(0)}}{dt} + \alpha_k \psi_k^{(0)} + \beta_k \psi_k^{(1)} = p, \quad \frac{d\psi_k^{(1)}}{dt} + \alpha_k \psi_k^{(1)} - \beta_k \psi_k^{(0)} = 0. \tag{6}$$

The boundary conditions can then be formulated by inserting the velocity v_n calculated in Eq. (5) into the pressure term of the linear coupled wave equation $\partial p/\partial \mathbf{n} = -\rho_0(\partial v_n/\partial t)$.

2.3 PINNs

Two multi-layer feed-forward neural networks are setup,

$$\hat{f} : (x, t, x_0) \mapsto \mathcal{N}_f(x, t, x_0; \mathbf{W}, \mathbf{b}), \quad \hat{g} : (x, t, x_0) \mapsto \mathcal{N}_{\text{ADE}}(x, t, x_0; \mathbf{W}, \mathbf{b}),$$

where $\mathbf{W}$ and $\mathbf{b}$ are the network weights and biases, respectively; $\mathcal{N}_{\text{ADE}}$ is only applied in case of frequency-dependent boundaries. The networks take three inputs x, t, x_0 corresponding to the spatial, temporal, and source position dimensions. The network $\mathcal{N}_f$ has one output $\hat{p}(x, t, x_0)$ approximating $p(x, t, x_0)$ predicting the pressure; the network $\mathcal{N}_{\text{ADE}}$ is a multi-output network with the number of outputs corresponding to the number of accumulators to approximate as explained in the following.

Including only information about the underlying physics, the governing partial differential equation (PDE) from Eq. (1) and initial conditions (ICs) from Eq. (2) can be learned by minimizing the mean squared error loss denoted $\|\bullet\|$ as

$$\underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}} \mathcal{L}(\mathbf{W}, \mathbf{b}) = \mathcal{L}_{\text{PDE}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{ADE}}, \tag{7}$$

where

$$\mathcal{L}_{\text{PDE}} = \left\| \frac{\partial^2}{\partial t^2}\mathcal{N}_f(x_f^i, t_f^i, x_{0,f}^i; \mathbf{W}, \mathbf{b}) - c^2\nabla^2\mathcal{N}_f(x_f^i, t_f^i, x_{0,f}^i; \mathbf{W}, \mathbf{b}) \right\|, \tag{8a}$$

$$\mathcal{L}_{\text{IC}} = \left\| \mathcal{N}_f(x_{\text{ic}}^i, 0, x_{0,\text{ic}}^i; \mathbf{W}, \mathbf{b}) - \exp\left[-\left(\frac{x_{\text{ic}}^i - x_{0,\text{ic}}^i}{\sigma_0} \right)^2 \right] \right\| + \left\| \frac{\partial}{\partial t}\mathcal{N}_f(x_{\text{ic}}^i, 0, x_{0,\text{ic}}^i; \mathbf{W}, \mathbf{b}) \right\|. \tag{8b}$$

Here, $\{x_{\text{ic}}^i, x_{0,\text{ic}}^i\}_{i=1}^{N_{\text{IC}}}$ denotes the initial N_{IC} data points, $\{x_f^i, t_f^i, x_{0,f}^i\}_{i=1}^{N_f}$ denotes the N_f collocation points for the PDE f, and the penalty weights λ_{IC} and λ_{BC} are used for balancing the impact of the individual terms. The loss function $\mathcal{L}_{\text{BC}}$ will be treated separately for the impedance boundary conditions in the following, where $\{x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i\}_{i=1}^{N_{\text{BC}}}$ will, correspondingly, be denoting the N_{BC} collocation points on the boundaries. For frequency-dependent boundaries, an auxiliary neural network will be coupled, resulting in the additional loss term $\mathcal{L}_{\text{ADE}}$ in Eq. (7) and is explained in detail in the following.

2.3.1 Frequency-independent impedance boundary loss functions

The frequency-independent boundary condition, Eq. (3), is included in the loss function $\mathcal{L}_{\text{BC}} := \mathcal{L}_{\text{INDEP}}$ satisfied by $\mathcal{L}_{\text{INDEP}} = \|(\partial/\partial t)\mathcal{N}_f(x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i; \mathbf{W}, \mathbf{b}) + c\xi(\partial/\partial \mathbf{n})\mathcal{N}_f(x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i; \mathbf{W}, \mathbf{b})\|$.

2.3.2 Frequency-dependent impedance boundary loss functions

For frequency-dependent boundaries, the ADEs need to be solved as well, approximating the ODEs from Eq. (6) by introducing an additional neural network $\mathcal{N}_{\text{ADE}}(x_b, t, x_0; \mathbf{W}, \mathbf{b})$ parameterized by x_b and x_0 for boundary positions and moving sources, respectively. The network has multiple outputs $\mathcal{N}_{\text{ADE}}(x_b, t, x_0; \mathbf{W}, \mathbf{b}) = [\tilde{\phi}_0, \tilde{\phi}_1, \ldots, \tilde{\phi}_{Q-1}; \tilde{\psi}_0^{(0)}, \tilde{\psi}_1^{(0)}, \ldots, \tilde{\psi}_{S-1}^{(0)}; \tilde{\psi}_0^{(1)}, \tilde{\psi}_1^{(1)}, \ldots, \tilde{\psi}_{S-1}^{(1)}]$ corresponding to the *scaled* accumulators determined by a scaling factor $l_{\text{ADE}}^{\bullet}$, mapping $\tilde{\phi}_k = l_{\text{ADE}}^{\phi_k}\hat{\phi}_k$, $\tilde{\psi}_k^{(0)} = l_{\text{ADE}}^{\psi_k}\hat{\psi}_k^{(0)}$, and $\tilde{\psi}_k^{(1)} = l_{\text{ADE}}^{\psi_k}\hat{\psi}_k^{(1)}$, such that $\tilde{\phi}_k, \tilde{\psi}_k^{(0)}, \tilde{\psi}_k^{(1)} : (x_b, t, x_0) \mapsto [-1, 1]$ matching the range of the tanh function used in this work. The scaling factors are independent of the geometry and domain dimensionality, only the material properties determine the amplitude of the accumulators. In this work, the scaling factors are determined using the SEM solver, but might be analytically estimated from the accumulators considering only a single reflection in a 1D domain. A graphical representation of the neural network architectures for approximating the governing physical equations and ADEs is depicted in Fig. 1. The accumulators with parametrized moving sources and boundary positions $\tilde{\phi}_k(x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i)$, $\tilde{\psi}_k^{(0)}(x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i)$, $\tilde{\psi}_k^{(1)}(x_{\text{bc}}^i, t_{\text{bc}}^i, x_{0,\text{bc}}^i)$ can be learned by minimizing the mean squared error loss as (arguments omitted)

$$\underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}} \mathcal{L}_{\text{ADE}}(\mathbf{W}, \mathbf{b}) = \sum_k^{Q-1} \tilde{\lambda}_{\text{ADE}}^{\tilde{\phi}_k}\mathcal{L}_{\tilde{\phi}_k} + \sum_k^{S-1}\left(\tilde{\lambda}_{\text{ADE}}^{\tilde{\psi}_k^{(0)}}\mathcal{L}_{\tilde{\psi}_k^{(0)}} + \tilde{\lambda}_{\text{ADE}}^{\tilde{\psi}_k^{(1)}}\mathcal{L}_{\tilde{\psi}_k^{(1)}} \right), \tag{9}$$

where

$$\mathcal{L}_{\tilde{\phi}_k} = \left\| \frac{\partial}{\partial t}\tilde{\phi}_k + \lambda_k\tilde{\phi}_k - l_{\text{ADE}}^{\phi_k}\mathcal{N}_f \right\|, \tag{10a}$$

$$\mathcal{L}_{\tilde{\psi}_k^{(0)}} = \left\| \frac{\partial}{\partial t}\tilde{\psi}_k^{(0)} + \alpha_k\tilde{\psi}_k^{(0)} + \beta_k l_{\text{ADE}}^{\psi_k^{(0)}}(1/l_{\text{ADE}}^{\psi_k^{(1)}})\tilde{\psi}_k^{(1)} - l_{\text{ADE}}^{\psi_k^{(0)}}\mathcal{N}_f \right\|, \tag{10b}$$

$$\mathcal{L}_{\tilde{\psi}_k^{(1)}} = \left\| \frac{\partial}{\partial t}\tilde{\psi}_k^{(1)} + \alpha_k\tilde{\psi}_k^{(1)} - \beta_k l_{\text{ADE}}^{\psi_k^{(1)}}(1/l_{\text{ADE}}^{\psi_k^{(0)}})\tilde{\psi}_k^{(0)} \right\|, \tag{10c}$$

and

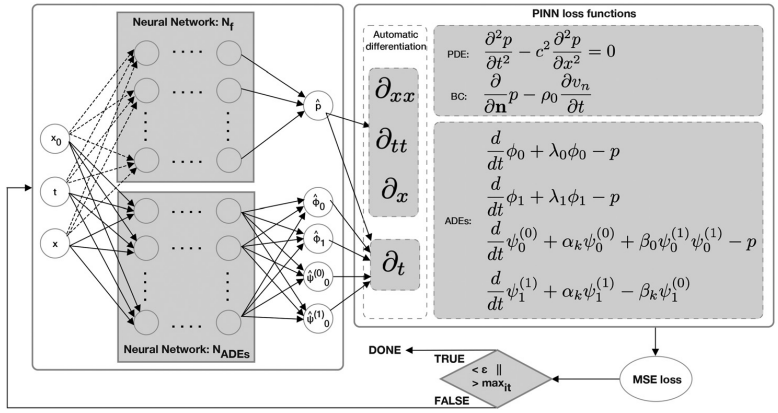


Fig. 1. PINN scheme for frequency-dependent boundaries. Left: Two fully connected feed-forward neural network architectures, $\mathcal{N}_f$ (PDE + ICs) and $\mathcal{N}_{\text{ADE}}$ (ADEs). Right: The governing physical equations and ADEs are coupled via the loss function (ICs and scaling terms are omitted for brevity). Training is done when a maximum number of epochs is reached, or the total loss is smaller than a given threshold.

$$\tilde{\lambda}^{\tilde{\phi}_k}_{\text{ADE}} = (1/l^{\phi_k}_{\text{ADE}})\lambda^{\tilde{\phi}_k}_{\text{ADE}}, \quad \tilde{\lambda}^{\tilde{\psi}^{(0)}_k}_{\text{ADE}} = (1/l^{\psi^{(0)}_k}_{\text{ADE}})\lambda^{\tilde{\psi}^{(0)}_k}_{\text{ADE}}, \quad \tilde{\lambda}^{\tilde{\psi}^{(1)}_k}_{\text{ADE}} = (1/l^{\psi^{(1)}_k}_{\text{ADE}})\lambda^{\tilde{\psi}^{(1)}_k}_{\text{ADE}}.$$

The frequency-dependent boundary conditions are satisfied by $\mathcal{L}_{\text{DEP}} = \|(\partial/\partial\mathbf{n})\mathcal{N}_f(x^i_{\text{bc}}, t^i_{\text{bc}}, x^i_{0,\text{bc}}; \mathbf{W}, \mathbf{b}) + \rho_0[\partial v_n(x^i_{\text{bc}}, t^i_{\text{bc}}, x^i_{0,\text{bc}})/\partial t]\|$, where v_n is the expression at the boundaries given in Eq. (5) with $\hat{\phi}_k = 1/l^{\phi_k}_{\text{ADE}}\tilde{\phi}_k$, $\hat{\psi}^{(0)}_k = 1/l^{\psi^{(0)}_k}_{\text{ADE}}\tilde{\psi}^{(0)}_k$, and $\hat{\psi}^{(1)}_k = 1/l^{\psi^{(1)}_k}_{\text{ADE}}\tilde{\psi}^{(1)}_k$, ensuring that the accumulators are properly re-scaled. The loss is included as the term $\mathcal{L}_{\text{BC}} := \mathcal{L}_{\text{DEP}}$ in Eq. (7) together with the loss for the ADEs in Eq. (9).

2.4 Setup

TENSORFLOW 2.5.1,[18] SCIANN 0.6.4.7,[19] and PYTHON 3.8.9 with 64 bit floating points for the neural network weights are used. The code for reproducing the results can be found online.[20]

Reference data for impedance boundaries are generated using a fourth-order Jacobi polynomial SEM solver. The grid was discretized with 20 points per wavelength spanning frequencies up to 1000 Hz yielding an average grid resolution of $\Delta x = 0.017$ m, and the time step was $\Delta t = \text{CFL} \times \Delta x/c$, where CFL is the Courant-Friedrichs-Lewy constant (CFL = 1.0 and CFL = 0.1 for frequency-independent and dependent boundaries, respectively). The speed of sound $c = 1$ m/s is used for the PINN setup, implying $\Delta x = \Delta t/c = \Delta t$, which is a normalization introduced to ensure the same scaling in time and space required for the optimization problem to converge. In case of a normalized speed of sound, the effective normalized frequency is correspondingly $f = f_{\text{phys}}/c_{\text{phys}}$, since the wave is now travelling slower compared to the physical setup. To evaluate the results for a physical speed of sound $c_{\text{phys}} = 343$ m/s, the temporal dimension should be converted back as $t_{\text{phys}} = t/c_{\text{phys}}$ s. In case of frequency-dependent boundaries, the velocity from Eq. (5) needs to be normalized accordingly regarding frequency and flow resistivity $\sigma_{\text{mat}} = \sigma_{\text{mat, phys}}/c_{\text{phys}}$ in Miki's model. Fitting the parameters for $c = 1$ m/s yields modified λ_k, α_k, β_k, and Y_∞ values resulting in the exact same surface impedance as for $c = 343$ m/s, but scaled by c_{phys} in frequency range and amplitude. This can be seen from the complex wavenumber and characteristic impedance of the porous medium in Miki's model involving f/σ_{mat} and $2\pi f/c$ not being affected by normalization.[20]

The point distribution in time and space, number of sources, penalty weights λ and scaling factors l_{ADE} for the ADEs are listed in Table 1. Note that the number of (time and space) domain points (30%) per source (7) is $47\,089 \times 0.3/7 = 2018$, satisfying the Nyquist theorem, since $\Delta x = 2/\sqrt{2018} = 0.045$ m (we can use the square root to get the gridpoint distribution in the spatial dimension) resulting in ppw $= \lambda_w/\Delta x = 7.6$ points per wavelength; $\lambda_w = c/f$

Table 1. Number of points in time and space; inner domain, boundaries, and initial condition point distributions; number of evenly distributed sources (srcs); values for the penalty weights λ; scaling factors l for normalizing the accumulators.

#total	#BC	#IC	#inner[a]	#srcs	λ_{IC}	λ_{BC}	$\lambda^{\bullet}_{\text{ADE}}$	$l^{\phi_0}_{\text{ADE}}$	$l^{\phi_1}_{\text{ADE}}$	$l^{\psi^{(0)}_0}_{\text{ADE}}$	$l^{\psi^{(1)}_0}_{\text{ADE}}$
47, 089	45%	25%	30%	7	20	1	10	10.3	261.4	45.9	22.0

[a]The centered Latin hypercube sampling strategy (Ref. 24) is used.

being the wavelength for physical frequency 1000 Hz and physical speed of sound 343 m/s. For the neural network, we have used the ADAM optimizer and the mean-squared error for calculating the losses for both networks. The training was run with learning rate 1e−4 and batch size 512 until a total loss of $\epsilon = 2e-4$ was reached (roughly 16k and 20k epochs needed for frequency-independent and dependent boundaries, respectively). The relatively big batch size was chosen to ensure that enough initial and boundary points were included in the optimization steps.

The network architecture of $\mathcal{N}_f$ consists of three layers, each with 256 neurons applying the sine activation function in each layer except for a linear output layer, with proper weight initialization.[21] Using sine activation functions can be seen as representing the signal using Fourier series[22] and is probably the reason for a significantly better convergence compared to using the more common choice of tanh activation functions. However, experiments showed degraded interpolation properties using sine activation functions when the network was trained on grids with source positions distributed more sparsely (0.3 m), even when lowering the number of neurons to prevent overfitting. A reason could be related to the distributed source interval violating the Nyquist sampling theorem $\Delta x < c/(2f) = 0.17$ m and causing aliasing effects, but this remains an open question. Therefore, the source positions were distributed evenly with finer resolution to improve the results between the source positions, consequently resulting in a sparser grid per source by keeping the total number of points the same. Despite the sparser grid, the convergence and final error still showed satisfying results. Distributing the source positions more densely is trivial in a data-free implementation, but if a combination of the underlying physics and simulated/measured data is considered later, a large number of source positions could be practically challenging.

The network architecture of $\mathcal{N}_{\text{ADE}}$ consists of three layers, each with 20 neurons applying the tanh activation function in each layer except for a linear output layer, with Glorot normal initialization of the weights.[23] Using the tanh function is an obvious choice since we have chosen to scale the accumulators to take values in the range $[-1, 1]$.

3. Results

Frequency-independent and dependent boundary conditions are tested, each with parameterized moving sources trained at seven evenly distributed positions $\mathbf{x}_0 = [-0.3, -0.2, \ldots, 0.3]$ m and evaluated at five positions $\mathbf{x}_0 = [-0.3, -0.15, 0.0, 0.15, 0.3]$ m. Additional results are included as supplementary materials.[20] The source is satisfied through the initial

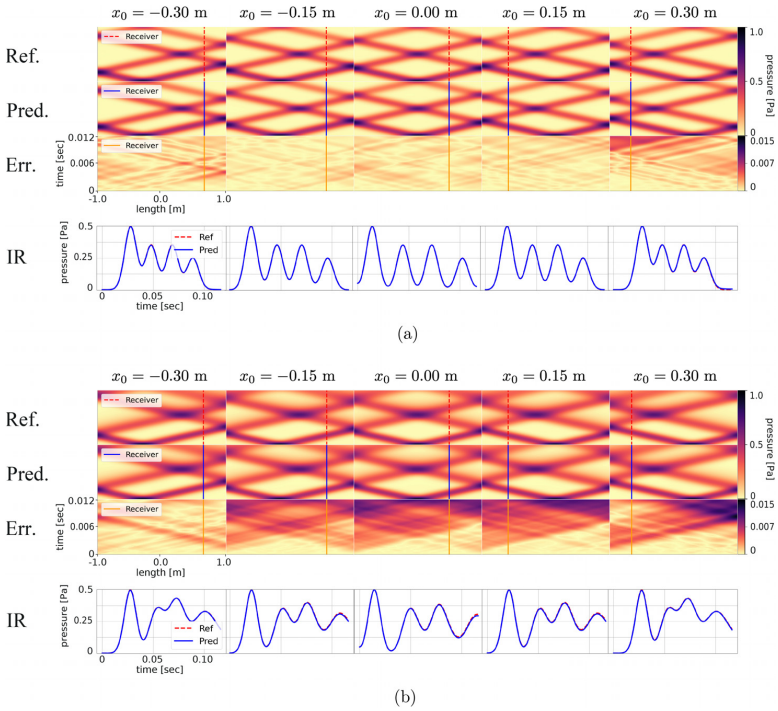


Fig. 2. Wave propagations in a 1D domain $[-1, 1]$ m.

Table 2. Time domain errors for source/receiver pairs (x_0, x) measured in meters in what follows at positions $s_0 = (-0.3, 0.64)$, $s_1 = (-0.15, 0.58)$, $s_2 = (0.0, 0.5)$, $s_3 = (0.15, -0.58)$, $s_4 = (0.3, -0.66)$ $\mu_{\text{rel}}(x, x_0) = (1/N)\sum_{i=0}^{N-1}[|\hat{p}(x, t^i, x_0) - p(x, t^i, x_0)|/p(x, t^i, x_0)]$ is the relative mean error over time within −60 dB range and $\infty_{\text{abs}}(x, x_0) = \max\{|\hat{p}(x, t^i, x_0) - p(x, t^i, x_0)| : i = 0 \cdots N-1\}$ Pa is the maximum absolute error for source/receiver pair s_i.

	s_0		s_1		s_2		s_3		s_4	
	μ_{rel}	∞_{abs}	μ_{rel}	∞_{abs}	μ_{rel}	∞_{abs}	μ_{rel}	∞_{abs}	μ_{rel}	∞_{abs}
Freq. indep.	0.7%	0.004	0.3%	0.002	0.5%	0.001	0.4%	0.002	1.5%	0.006
Freq. dep.	0.5%	0.004	1.5%	0.008	1.7%	0.011	1.9%	0.009	1.1%	0.005

condition modeled as a Gaussian impulse from Eq. (2) with $\sigma_0 = 0.2$ spanning frequencies up to 1000 Hz. The speed of sound $c_{\text{phys}} = 343$ m/s and air density $\rho_0 = 1.2\,\text{kg/m}^3$ are used for all studies.

First, we test the frequency-independent boundary condition with normalized impedance $\xi = 5.83$ depicted in Fig. 2(a). Frequency-independent boundaries with corresponding wave propagation animations available from Mm. 1. Then, we test the frequency-dependent impedance boundary condition, where the boundary is modeled as a porous material mounted on a rigid backing with thickness $d_{\text{mat}} = 0.10$ m with an air flow resistivity of $\sigma_{\text{mat,phys}} = 8000\,\text{Nsm}^{-4}$. The surface impedance Y of this material is estimated using Miki's model[25] and mapped to a two-pole rational function in the form of Eq. (4) with $Q = 2$ and $S = 1$ using a vector fitting algorithm[26] yielding the coefficients for Eq. (5). The results are depicted in Fig. 2(b). Frequency-dependent boundaries with corresponding wave propagation animations available from Mm. 2.

Mm. 1. Frequency-independent boundaries, animation. Same parameters as Fig. 2(a). Frequency-independent boundaries. File of type "mp4" (1.3 MB).

Mm. 2. Frequency-dependent boundaries, animation. Same parameters as Fig. 2(b). Frequency-dependent boundaries. File of type "mp4" (1.6 MB).

We observe that the shape of the wave propagations is well captured, and the impulse responses also fit the reference solutions very well for all boundary types. The relative mean error $\mu_{\text{rel}}(x, x_0) = (1/N)\sum_{i=0}^{N-1}[|\hat{p}(x, t^i, x_0) - p(x, t^i, x_0)|/p(x, t^i, x_0)]$ within −60 dB and absolute maximum error $\infty_{\text{abs}}(x, x_0) = \max\{|\hat{p}(x, t^i, x_0) - p(x, t^i, x_0)| : i = 0 \cdots N-1\}$ of the impulse responses originating from various source and receiver positions are summarized in Table 2. Relative errors are below 2%/0.2 dB for all predictions. The absolute maximum errors are below 0.011 Pa for all predictions indicating that no severe outliers are present.

4. Conclusion and future work

A novel method is presented for predicting the sound field in a 1D domain for impedance boundaries and parameterized moving Gaussian sources using PINNs. A coupled system of differential equations, consisting of the governing physical equations and a system of ODEs predicting the accumulators of the ADEs, was used for training the PINN. The equations for the ADEs depend only on time t but were parameterized to take boundary and source positions into account, yielding a very flexible implementation. The results are promising, with relative mean errors below 2%/0.2 dB for all cases. The approach taken by learning a compact surrogate model that is inexpensive to evaluate at runtime shows potential to overcome current numerical methods' limitations in modeling flexible scenes, such as moving sources.

Compared to standard numerical methods, the PINN method takes up to three orders of magnitude more time to converge. Therefore, to solve realistic problems in 3D, the convergence rate needs to be improved. This is partly due to the need for fairly large amounts of grid points with 70% of the points located at the initial time step and at boundaries where penalty weights are also needed for balancing each loss term. Formulating an ansatz imposing initial and boundary conditions directly could overcome this problem.[27] Also, considering other architectures taking (discrete) time-dependence into account instead of optimizing the entire spatiotemporal domain at once might improve the learning rate and produce more precise results. Moreover, we have observed challenges in the global optimizer for a larger domain size and/or by increasing the frequency due to the ratio between zero and non-zero pressure values. Domain decomposition methods[28] have been introduced to overcome this limitation. In ongoing work, more complex benchmarks are being considered.

Acknowledgments

Thanks to DTU Computing Center GPULAB for access to GPU clusters and swift help. Also, a big thanks to Ehsan Haghighat for valuable discussions regarding PINNs and SciANN. Last but not least, thanks to Finnur Pind for making an SEM code available for calculating reference solutions.

References and links

[1]T. Okuzono, T. Otsuru, R. Tomiku, and N. Okamoto, "A finite-element method using dispersion reduced spline elements for room acoustics simulation," Appl. Acoust. **79**, 1–8 (2014).

[2]F. Pind, A. P. Engsig-Karup, C.-H. Jeong, J. S. Hesthaven, M. S. Mejling, and J. Strømann-Andersen, "Time domain room acoustic simulations using the spectral element method," J. Acoust. Soc. Am. **145**(6), 3299–3310 (2019).
[3]A. Melander, E. Strøm, F. Pind, A. Engsig-Karup, C.-H. Jeong, T. Warburton, N. Chalmers, and J. S. Hesthaven, "Massive parallel nodal discontinuous Galerkin finite element method simulator for room acoustics," Int. J. High Perform. Comput. Appl. (2020), http://infoscience.epfl.ch/record/279868 (Last viewed 30/10/2021).
[4]D. Botteldoorena, "Finite-difference time-domain simulation of low-frequency room acoustic problems," J. Acoust. Soc. Am. **98**(6), 3302–3308 (1995).
[5]B. Hamilton and S. Bilbao, "FDTD methods for 3-D room acoustics simulation with high-order accuracy in space and time," IEEE/ACM Trans. Audio Speech Lang. Proc. **25**(11), 2112–2124 (2017).
[6]N. Raghuvanshi and J. Snyder, "Parametric wave field coding for precomputed sound propagation," ACM Trans. Graph. **33**(4), 1 (2014).
[7]N. Raghuvanshi, "Dynamic portal occlusion for precomputed interactive sound propagation," arXiv:2107.11548 (2021).
[8]J. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods Parametrized Partial Differential Equations* (Springer, Berlin, 2015) pp. 1–131.
[9]H. S. Llopis, A. P. Engsig-Karup, C.-H. Jeong, F. Pind, and J. S. Hesthaven, "Efficient numerical room acoustic simulations with parametrized boundaries using the spectral element and reduced basis method," arXiv:2103.11730 (2021).
[10]D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," AIChE J. **38**(10), 1499–1511 (1992).
[11]I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Networks **9**(5), 987–1000 (1998).
[12]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[13]Z. Fan, V. Vineet, H. Gamper, and N. Raghuvanshi, "Fast acoustic scattering using convolutional neural networks," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings 2020-May* (2020), pp. 171–175.
[14]B. Moseley, A. Markham, and T. Nissen-Meyer, "Solving the wave equation with physics-informed deep learning," arXiv:2006.11894 (2020).
[15]M. Rasht-Behesht, C. Huber, K. Shukla, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions," arXiv:2108.12035 (2021), pp. 1–29.
[16]H. Kuttruff, *Room Acoustics*, 6th ed. (CRC Press, Boca Raton, 2016), p. 322.
[17]R. Troian, D. Dragna, C. Bailly, and M. A. Galland, "Broadband liner impedance education for multimodal acoustic propagation in the presence of a mean flow," J. Sound Vib. **392**, 200–216 (2017).
[18]Google, "TensorFlow" (2021), https://www.tensorflow.org/ (Last viewed 30/10/2021).
[19]E. Haghighat and R. Juanes, "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks," Comput. Methods Appl. Mech. Eng. **373**, 113552 (2021).
[20]See supplementary material at https://www.scitation.org/doi/suppl/10.1121/10.0009057 for source code and addition results for Neumann boundaries, accumulator predictions, runtime efficiency of the surrogate model, and detailed explanation of the normalization for the frequency-dependent impedance boundary formulation.
[21]V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," arXiv:2006.09661 (2020).
[22]N. Benbarka, T. Höfer, H. ul-moqeet Riaz, and A. Zell, "Seeing implicit neural representations as fourier series," arXiv:2109.00249 (2021).
[23]X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9 of Proceedings of Machine Learning Research*, edited by Y. W. Teh and M. Titterington, PMLR, Chia Laguna Resort, Sardinia, Italy (2010), pp. 249–256.
[24]M. Stein, "Large sample properties of simulations using Latin hypercube sampling," Technometrics **29**(2), 143–151 (1987).
[25]Y. Miki, "Acoustical properties of porous materials-modifications of Delany-Bazley models" J. Acoust. Soc. Jpn. (E) **11**(1), 19–24 (1990).
[26]B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by vector fitting," IEEE Trans. Power Deliv. **14**(3), 1052–1061 (1999).
[27]N. Sukumar and A. Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks," arxiv.org/abs/2104.08426 (2021).
[28]K. Shukla, A. D. Jagtap, and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition," arxiv.org/abs/2104.10013 (2021).

Paper B

Sound propagation in realistic interactive 3D scenes with parameterized sources using deep neural operators

Nikolas Borrel-Jensen[a], Somdatta Goswami[c], Allan P. Engsig-Karup[b], George Em Karniadakis[c,d], Cheol-Ho Jeong[a,*]

[a]*Department of Electrical and Photonics Engineering, Acoustic Technology, Technical University of Denmark, Ørsteds Plads, 2800 Kgs. Lyngby, Denmark*
[b]*Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, 2800 Kgs. Lyngby, Denmark*
[c]*Division of Applied Mathematics, Brown University, 170 Hope Street, Providence, RI - 02906, U.S.A.*
[d]*School of Engineering, Brown University, 170 Hope Street, Providence, RI - 02906, U.S.A.*

Abstract

We address the challenge of acoustic simulations in 3D virtual rooms with parametric source positions, which have applications in virtual/augmented reality, game audio, and spatial computing. The wave equation can fully describe wave phenomena such as diffraction and interference. However, conventional numerical discretization methods are computationally expensive when simulating hundreds of source and receiver positions, making simulations with parametric source positions impractical. To overcome this limitation, we propose using deep operator networks to approximate linear wave-equation operators. This enables the rapid prediction of sound propagation in realistic 3D acoustic scenes with parametric source positions, achieving millisecond-scale computations. By learning a compact surrogate model, we avoid the offline calculation and storage of impulse responses for all relevant source/listener pairs. Our experiments, including various complex scene geometries, show good agreement with reference solutions, with root mean squared errors ranging from 0.02 Pa to 0.10 Pa. Notably, our method signifies a paradigm shift as – to our knowledge – no prior machine learning approach has achieved precise predictions of complete wave fields within realistic domains.

Keywords: Virtual acoustics, Operator learning, DeepONet, Transfer learning, Domain decomposition

1. Introduction

Wave phenomena are precisely described by solving partial differential equations (PDEs) with their approximate solutions found using numerical methods. Many methods exist, such as finite-difference time-domain methods (FDTD) [1], finite-volume time-domain methods (FVTD) [2], finite/spectral element methods (SEM) [3], discontinuous Galerkin methods (DG-FEM) [4], boundary element methods (BEM) [5], and pseudo-spectral Fourier methods [6], and are all part of the standard toolbox used to successfully solve a variety of real-world

*Corresponding author.

physical problems over the last decades. Determining which method to use depends on the nature and difficulty of the problem, geometric complexity, and trade-offs between accuracy and efficiency. However, all these methods require recalculating solutions for different conditions, including initial and boundary conditions, geometry, and specified source and receiver positions. Obtaining a solution even on a 2D domain is often computationally expensive; hence, solving parameterized PDEs involving multiple parameters or varying conditions can quickly get intractable.

In this work, we address the challenges in solving the wave equation for the four geometries depicted in Figure 1 considering its relevance in virtual acoustics, which plays a pivotal role in computer games, mixed reality, and spatial computing [7]. Creating a realistic auditory environment in these applications is crucial for an immersive user experience. The impulse responses (IR) characterizing the room's acoustical properties for a source/receiver pair can be obtained using the numerical methods referenced at the beginning of the section. This is done offline for real-time applications due to the computational requirements, especially when spanning a broad frequency range. However, for dynamic, interactive scenes with numerous parametric source and receiver pairs, the storage requirement for a lookup database becomes intractable (in the gigabytes range). These challenges become even more extensive when covering the audible frequency range up to 20 kHz. Employing surrogate models to learn the parametrized solutions to the wave equation to obtain a one-shot continuous wave propagation in interactive scenes [8] offers an ideal framework to address the prevailing challenges in virtual acoustics applications, effectively surpassing the limitations of traditional numerical methods.

The idea of approximating continuous nonlinear operators for parametrized PDEs from labeled data was first introduced in 1995 by Chen & Chen [9] providing a universal operator approximation theorem for shallow neural networks, guaranteeing small approximation errors (the error between the target operator and the predictions from a class of infinitely wide neural network architectures). Recently in 2019, Lu et al. [10] reformulated Chen & Chen's theorem and generalized the work by proposing the deep operator network architecture 'DeepONet,' which exhibits small generalization errors (the ability of a neural network to produce small errors for unseen data). Acknowledging the previous successful application of DeepONet in fracture mechanics [11], diesel engine [12], microstructure evolution [13], bubble dynamics [14], bio-mechanics to detect aortic aneurysm [15] and airfoil shape optimization [16], to name a few, we consider this to be a suitable candidate for our problem.

Despite being a simple non-stiff second-order linear hyperbolic PDE, solving the wave equation is still challenging due to its multi-modal broadband-frequency nature. Therefore, learning a compact and efficient surrogate model to approximate the continuous operators of the wave equation emerges as a valuable solution for addressing a significant real-world challenge, such as virtual acoustics. The resulting DeepONet-based surrogate model should then: 1) predict the wave field propagation in rooms with parameterized sources and realistic frequency-dependent sources; 2) produce sufficiently accurate predictions for intended applications; and finally, 3) infer in real-time (< 100 ms). However, the predictive performance of DeepONet is often restricted by the availability of high-fidelity labeled datasets used for training. Moreover, undertaking isolated learning, which involves training a single predictive model for different yet related single tasks, can be exceedingly expensive. To mitigate this bottleneck, we have introduced a simple transfer-learning framework to transfer knowledge

between relevant domains [17]. The transfer learning framework allows the target model to be trained with limited labeled data to approximate solutions on a different but related domain, achieving the same accuracy as the source model, a model trained with a sufficiently labeled dataset on a specific domain. Finally, we push the boundaries of the seminal DeepONet to propose a domain-decomposition framework, which leverages the inherent property of deploying multiple deep neural networks in smaller subdomains, allowing for parallelization. Additionally, it is designed to handle large complex geometries, further expanding the applicability and scalability of the DeepONet method.

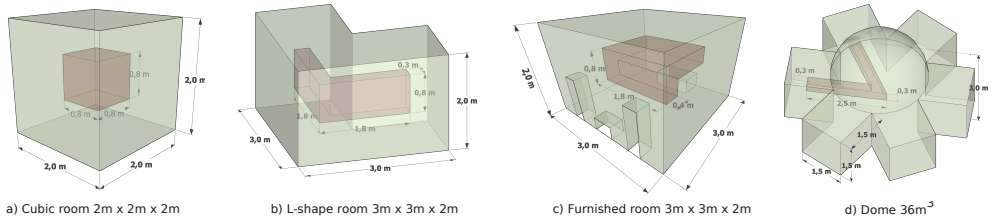


Figure 1: Pictorial representations of the domain geometries adopted in this work to evaluate the predicted 3D sound fields. All the experiments have parametric source positions allowed to move freely inside a subdomain of the room shown in shaded red.

2. Results

Four geometries, in increasing order of complexity, depicted in Figure 1, have been considered to evaluate the predicted 3D sound fields in *a*) a cubic 2 m × 2 m × 2 m room with frequency-dependent boundaries, *b*) an L-shape room with outer dimensions 3 m×3 m×2 m and frequency-dependent boundaries, *c*) a furnished room 3 m × 3 m × 2 m with frequency-dependent walls, ceiling and floor, and frequency-independent furniture, and *d*) a dome with a volume of 36 m^3 consisting of frequency-independent boundaries. For all the geometries, the models are learned through a final simulation time $T = 0.05$ seconds with parametric source positions allowed to move freely inside a sub-domain of the room shown in shaded red. The simulation time was chosen long enough to capture enough information to be meaningful and small enough to make the data generation and training time tractable. The impulse response consists of a direct sound followed by early reflections, which plays a key role in sound perception in rooms up to about 50-100 ms [18, 19] – slightly above the simulation time in this work. After the sound propagates over time, the response approaches decaying Gaussian noise, referred to as late reverberation. This part is known to be less crucial in sound perception and could be approximated by some statistical method [20].

Two experiments for the dome are performed; one where the model is trained for receiver positions in the full domain and another where the model is trained for receiver positions in 1/4 of the domain (denoted 'quarter model' in the rest of the manuscript); both cases allow for the source to move freely in the same subdomain. The quarter model applies a domain decomposition approach where separate DeepONets are trained on individual partitions for improved accuracy.

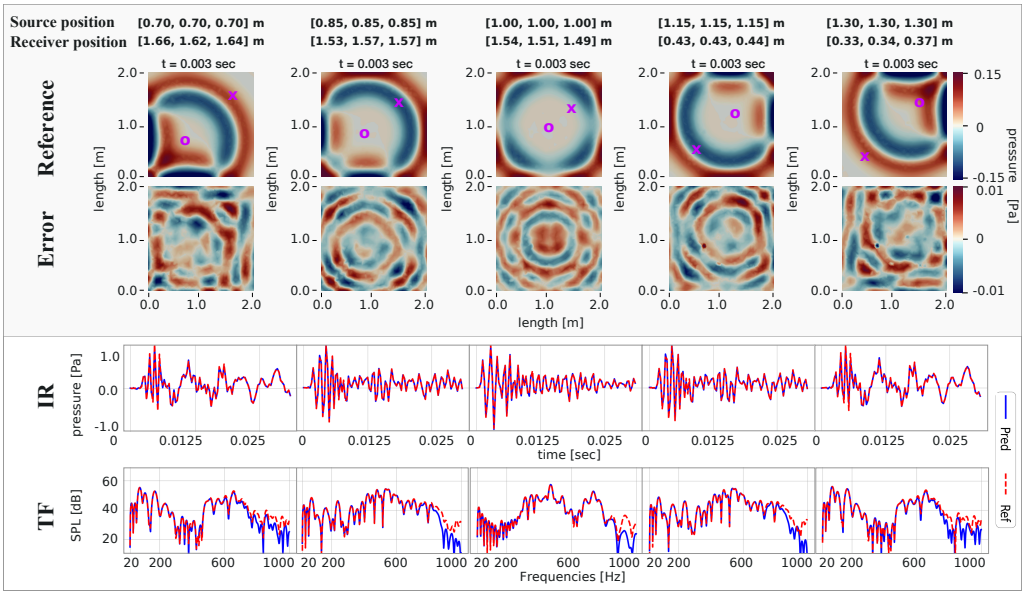


Figure 2: Cubic room $2 \times 2 \times 2$ m^3. Results show the sound field at $t = 0.003$ s for five parameterized source positions. The wave field error is depicted in the second row, and the IRs and TFs references and predictions are at the two bottom rows. 'o'=source position, 'x'=receiver position.

The training data has been generated using `ppw` $= 6$ points per wavelength, whereas validation and testing data has been generated using `ppw` $= 5$ to ensure (mostly) non-overlapping spatial samples to investigate the model's generalization capabilities; *i.e.,* how well the network interpolates at the receiver position, which is crucial for the applications of interest. The input function denoting a Gaussian pulse acting as an initial condition (Equation 2) is sampled at the Nyquist limit, whereas the density of the source positions is sampled at one-fifth of a wavelength for the training data, one full wavelength for the validation data, and five positions for the test data. Details about the data set and DeepONet network setup can be found in Materials and Methods. The data set sizes are summarized in Table B.1 ranging from $5.8\text{M} - 21.5\text{M}$ training samples, depending on the complexity of the geometry. The testing data is generated on the same grid as the validation data (different from the training data grid) but only for the five source/receiver pairs. Representative plots of the wave field reference and the corresponding error for the four geometries are presented in Figures 2-5. The plots also present the reference and prediction for the impulse response and the transfer function shown for each source/receiver pair. In Table B.2, the root mean square error (RMSE) for the IR is reported after performing $50 - 70$k iterations until saturation (Figure B.9).

2.1. Cubic room

Figure 2 shows an almost perfect fit between references and predictions and only minor differences in the upper-frequency range with a mean broadband RMSE of 0.03 Pa.

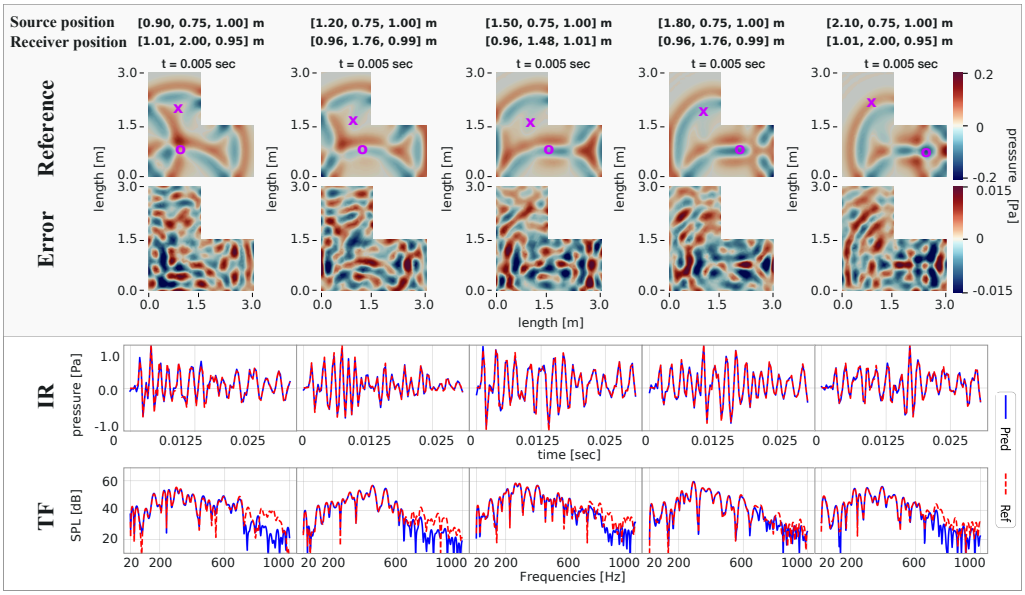


Figure 3: L-shape room with outer dimension $3 \times 3 \times 2$ m^3. Results show the sound field at $t = 0.005$ s for five parameterized source positions. The wave field error is depicted in the second row, and the IRs and TFs references and predictions are at the two bottom rows. 'o'=source position, 'x'=receiver position.

2.2. L-shape room

Similar to the previous example, in Figure 3, we see a good match between reference and prediction but with bigger deviations in the upper-frequency range above the 700–800 Hz limit. This deviation is also reflected in the mean RMSE of 0.05 Pa.

2.3. Furnished room

As shown in Figure 4, the wave propagation is well captured with quite good agreement between reference and prediction. Still, some inaccuracies are lacking for the sharp peaks, which can also be seen in the upper-frequency range above 600–700 Hz. The mean RMSE is 0.09 Pa, almost twice the error compared to the L-shape room and three times the error compared to the Cubic room.

2.4. Dome

The results for both the full and quarter models are evaluated at the same source and receiver positions for comparison shown in Figure 5. The receiver positions are restricted to the 1st quadrant where the quarter model was trained (denoted by the red square) and evaluated at five source/receiver pairs. The wave propagation is well captured for both the full and quarter models, with good agreement between the reference and the prediction. However, not all sharp peaks are well captured for the full model. The fit in the frequency domain is better than the furnished room but not quite as good as for the cubic and L-shape rooms, also indicated by the mean RMSE of 0.08 Pa. Applying domain decomposition and

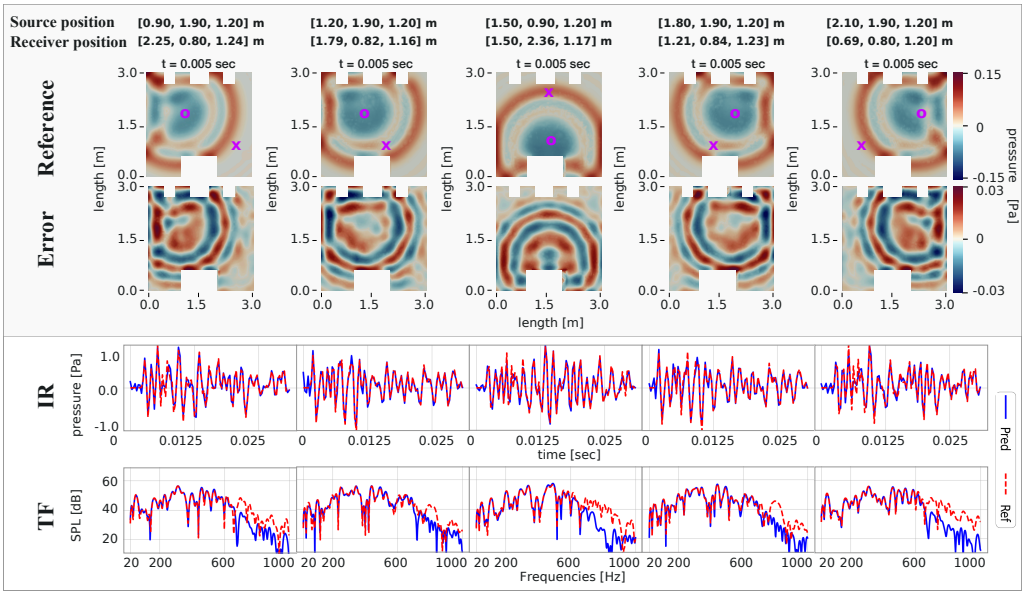


Figure 4: Furnished room $3 \times 3 \times 2$ m^3. Results show the sound field at $t = 0.005$ s for five parameterized source positions. The wave field error is depicted in the second row, and the IRs and TFs references and predictions are at the two bottom rows. 'o'=source position, 'x'=receiver position.

only training and evaluating the receivers in 1/4 of the domain gives significantly better results with a better fit in both the time and frequency domain, reporting a mean RMSE of 0.03 Pa on par with the cubic domain.

2.5. Run-time efficiency

For real-time applications, we assess the inference time of trained networks with identical layer and neuron configurations, except for the input layer of the branch net, which varies based on the geometry shape. This variation affects parameters and forward propagation performance. Summary information, including parameter count and total storage, is provided in Table B.3. Using an Nvidia V100 GPU, we predicted impulse responses of length $T = 0.5$ s (increased ten times compared to previous experiments), sampled at $f_s = 2000$ Hz, for five receiver positions. Execution times for the cubic, L-shape, furnished, and dome geometries were 39 ms, 49 ms, 49 ms, and 132 ms, respectively. Apart from the dome, these times comfortably meet the real-time threshold of 96 ms established by previous experiments [21]. Crossing this threshold would introduce significant degradation in azimuth error and elapsed time. The longer execution time for the dome is due to the larger input space covered by discretized input functions, spanning a larger volume compared to other geometries. Our DG-FEM data generation code constructs input function sizes based on the smallest enclosing bounding box and uniform distribution of samples, resulting in unused function values outside the dome geometry. Furthermore, the modified MLP network architecture expands the input layers, increasing the network size compared to a standard MLP network. Con-

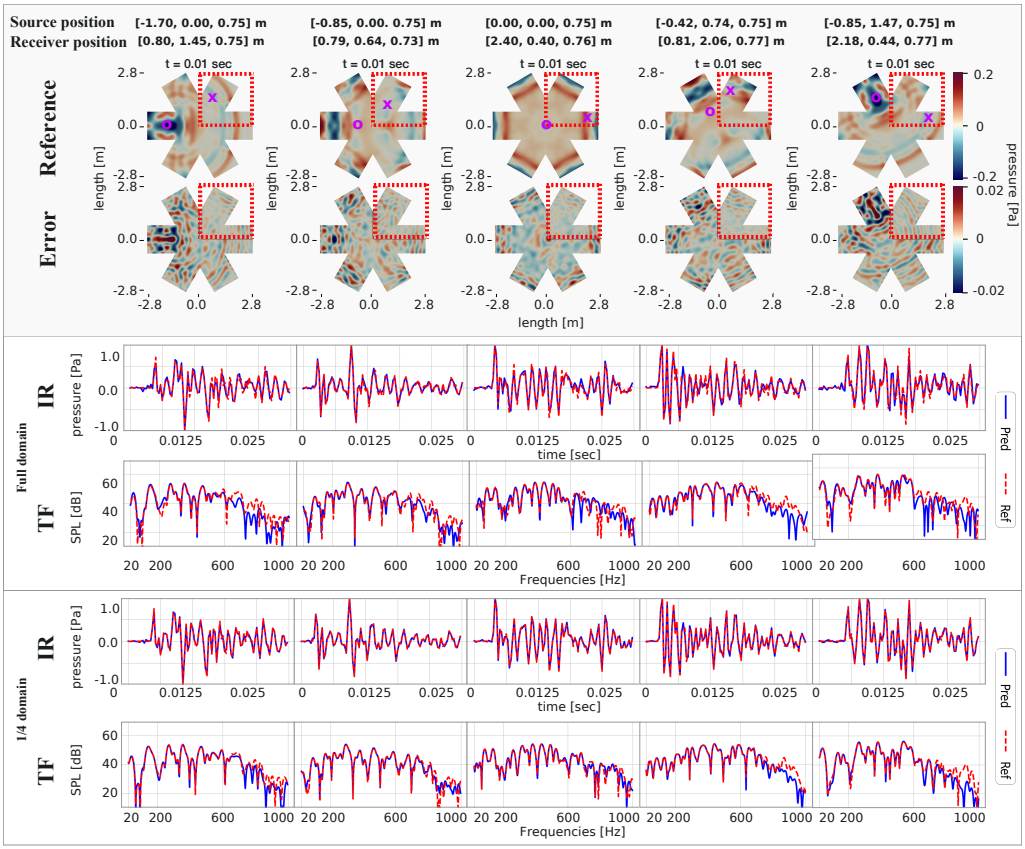


Figure 5: Dome 36 m^3. Results show the sound field at $t = 0.01$ s for five parameterized source positions. The IRs and TFs references and predictions are at the two bottom rows for the full and quarter partition. The red square denotes the receiver positions where the quarter model was trained.

volutional neural networks, as demonstrated in [22], offer comparable accuracy to modified MLPs while potentially enhancing inference speed.

2.6. Training time

Overall the training times for the 3D geometries are between one and three days on a single GPU. We divide the training time per iteration into data loading and weight/bias update encompassing forward/backward propagation. Our experiments were conducted in the 2D and 3D furnished rooms, as summarized in Table S4. Notably, training in 3D is approximately 64 times slower per iteration compared to 2D. In 2D, the data size of 229 MB fits in memory, while in 3D, the data size is 119 GB, necessitating streaming from disk[1]. This disparity is the primary reason for the significant increase in training time. Data

[1]960 GB SATA SSD connected to a node at the DTU Computing Center [23].

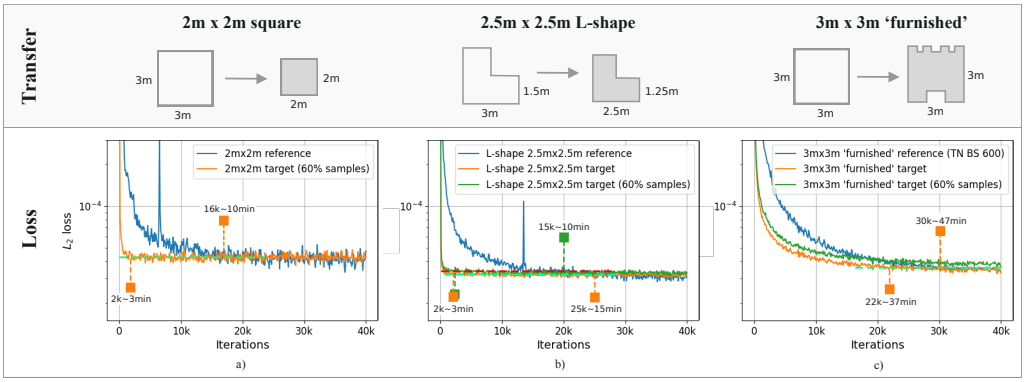


Figure 6: Comparative convergence plots of the reference model and the fine-tuned target model for transfer learning scenarios.

loading takes 2.1 seconds in 3D, while 2D (loaded from memory) only requires 32.7 ms. Consequently, the loading time is more than 1,200× longer in the 3D scenario. Additionally, the time for weights and biases update is 18× longer in 3D than in 2D due to the larger network size, while assuming similar accuracy, as both models in 2D and 3D exhibit a mean RMSE of 0.09 Pa.

2.7. Transfer learning

In Figure 6, the convergence rates for training a reference model from scratch and employing a well-trained source model to initialize the weights on a target model followed by fine-tuning are compared. Three cases are considered, *a*) a square 3×3 m^2 to a square 2×2 m^2, *b*) a square 3×3 m^2 to a furnished square 3×3 m^2, and *c*) an L-shape geometry with outer dimensions 3×3 m^2 to an L-shape geometry with outer dimensions 2.5×2.5 m^2. Significant improvements in training time are seen for cases *a*) and *b*), with a 3× speedup using only 60% of the data samples on the target domain. Mini-batching for the target model in transfer learning using the spatiotemporal batch size $Q = 600$ instead of $Q = 200$ for the source model nearly triples the training time but enhances the initial convergence rate, leading to sharper convergence. This effect could also be present in training the reference model, reaching the cross-over point sooner. However, the training time would increase beyond the time saved by a sooner cross-over, making this approach less effective. For example, the L-shape reference would cross at 16k iterations, taking 25 minutes using $Q = 600$, compared to 25k iterations, taking only 15 minutes using $Q = 200$. The convergence for training the 2×2 m^2 rectangular reference model would remain unchanged. In the case of the furnished shape, using the larger batch size narrows the performance gap between the reference and transfer learning in terms of both loss and time. Therefore, the reference model with the larger batch size is chosen for a fair comparison. Additionally, when utilizing only 60% of the samples, the convergence points for the reference and transfer models align earlier for the L-shape and furnished geometries.

3. Discussion

The results demonstrate good agreement between the prediction and reference for all five source/receiver pair positions, with RMSE values below 0.10 Pa. However, some inaccuracies are observed for sharp peaks and high-frequency content in both time and frequency domains, particularly in the furnished room and, to some extent, the full dome, primarily due to the large source position volume and domain volume, respectively. The cubic room shows the best result with a mean RMSE of 0.03 Pa, compared to 0.09 Pa for the furnished room, mainly due to the relatively small volume. The dome's source position subdomain has 1,849 source positions compared to $2{,}826 - 4{,}799$ source positions in the other geometries, which is considerably smaller. However, the predictions are less accurate primarily due to the large volume and, to a lesser extent, due to the geometrical complexity. To investigate the scalability of the DeepONet, instead of increasing the network sizes not necessarily yielding better accuracy/convergence, we applied a simple domain decomposition technique limiting the operator to be evaluated in 1/4 of the domain, still predicting for all 1,849 source positions. This technique showed much-improved accuracy on par with the cubic room, proposing a way to scale these methods to large-scale simulations. Applying the same technique to the furnished room should also increase its accuracy. However, it comes with the disadvantage of training additional DeepONets scaling with the number of partitions.

DeepONet more easily learns lower-frequency modes than higher-frequency modes. Using the `sine` activation function and employing Fourier expansion in the input layer to span multiple periods accomplishes the goal of helping the network learn higher-frequency content. Although the above modifications dramatically improved the learning capability of DeepONet, it still lacks some accuracy above 700 Hz for larger and more complex geometries. Increasing the number of layers and neurons did not improve the accuracy, indicating that the network bottleneck is not the capacity but rather difficulties in the optimization to find better optima. It is well-known that all neural networks have challenges when the ratio between upper-frequency and domain size gets larger, which is addressed in the current study by proposing a simple domain decomposition technique.

A spurious noise is observed in the impulse responses before the direct sound arrival, oscillating at the non-causal fundamental frequency. The network struggles to simultaneously learn wave propagation and zero pressures due to the trigonometric feature expansion and `sine` activation functions. This trade-off between aiding network learning with prior knowledge and learning zero pressures can be managed since the non-zero pressures are small and may not be audible in practical applications.

Training the 2D domain is efficient, taking less than 40 minutes, while training the 3D domains requires between one and three days, mainly due to streaming data from disk. This accounts for an overall 64× increase in time, with more than a 1,200× increase in data loading time for the furnished geometry. The larger network and batch sizes in 3D contribute to a 18× increase in training time, but the forward/backward propagation time scales better than the cubic complexity of standard numerical methods. Transfer learning experiments show a 3× speedup when fine-tuning the network parameters on scaled geometries for a square and an L-shape domain. Still, limited improvements are seen when transferring to a furnished domain. The results indicate that transfer learning frameworks could lead to faster training, provided that the source and target models are *similar enough* for the target

wave field to be learned efficiently.

The surrogate models exhibit efficient execution, with inference times below 49 ms for the cubic, L-shape, and furnished rooms, meeting real-time requirements for audio-visual applications. The dome's inference time is slower, taking 132 ms due to the larger dimensionality of the discretized source input functions. This could be addressed by sampling the input function more accurately, ensuring no zero-samples are outside the geometry. However, if the learned model is intended to be used as a source model for transfer learning, more sophisticated methods should be applied to relate spatial locations between models. Also, using convolutional neural networks for the branch net could decrease the number of network parameters.

To the authors' knowledge, this is the first time a surrogate model with parameterized source positions has been proposed for modeling wave propagation in 3D domains with realistic frequency-independent and dependent boundaries capable of executing in real-time. These findings are promising, with the potential to overcome current numerical methods' limitations in modeling flexible scenes, such as moving sources. However, further research is needed to address limitations related to larger rooms and better learning of high-frequency content when numerous degrees of freedom are required for source positions. Perceptual studies are also necessary to assess the tolerability of error levels for specific applications.

4. Governing equations

The acoustic wave equation for which a surrogate model is to be learned is given as

$$\frac{\partial^2 p(x,t)}{\partial t^2} - c^2 \frac{\partial^2 p(x,t)}{\partial x^2} = 0, \qquad t \in \mathbb{R}^+, \qquad x \in \mathbb{R}, \tag{1}$$

where p is the pressure (Pa), t is the time (s) and c is the speed of sound in air (m/s). The initial conditions (ICs) are satisfied by using a Gaussian impulse function (GIF) as sound source for the pressure part and setting the velocity equal to zero as

$$p(x, t=0, x_0) = \exp\left[-\left(\frac{x - x_0}{\sigma_0}\right)^2\right], \quad \frac{\partial p(x, t=0, x_0)}{\partial t} = 0, \tag{2}$$

with σ_0 being the width parameter of the pulse determining the frequencies to span (smaller σ_0 indicates a larger frequency span). The details concerning the boundary condition modeling can be found in Appendix B.2.

4.1. Code setup

JAX 0.4.10 [24], Flax 0.6.10 [25] and Python 3.10.7 have been used for all experiments and the code is available here: https://github.com/dtu-act/deeponet-acoustic-wave-prop.

4.1.1. Data

The physical speed of sound is $c_{\text{phys}} = 343$ m/s, and air density is $\rho_0 = 1.2$ kg/m^3, where the speed of sound has been normalized to $c = 1$ m/s to ensure the same resolution in the spatial and temporal dimensions. This is crucial for the optimizer performing gradient descent to find meaningful trajectories. The normalization of the speed of sound can be done trivially by adjusting the time as $t = t_{\text{phys}} \cdot c_{\text{phys}}$. Unless stated otherwise, the following

will present the results and material parameters in the physical domain. The frequency-independent boundaries are modeled with normalized impedance $\xi_{\text{imp}} = 17.98$, whereas the frequency-dependent boundaries are modeled as a porous material mounted on a rigid backing with thickness $d_{\text{mat}} = 0.03$ m with an airflow resistivity of $\sigma_{\text{mat}} = 10,000$ Nsm^{-4}. This material's surface impedance Y is estimated using Miki's model [26] and mapped to a six-pole rational function in the form given in Equation B.6 using a vector fitting algorithm [27] yielding the coefficients for the velocity term from Equation B.7.

The GIF to the branch network has been discretized at the Nyquist limit $\texttt{ppw} = 2$. Each sample corresponds to a specific source position, and the number of samples (i.e., the source density) needed for spanning the input space is calculated such that the average resolution between source positions is well-resolved w.r.t. the upper frequency, i.e., $\Delta x_{\text{source density}} = \frac{c}{f \cdot \text{ppw}}$.

The 3D data was generated using a DG-FEM solver [4], whereas the 2D data were generated using an SEM solver [3]. Ensuring good accuracy at interpolation locations is crucial for the applications of interest. Therefore, the training data was generated with $\texttt{ppw} = 6$ using six-order Jacobi polynomials for all cases except for the dome using fourth-order Jacobi polynomials. The validation and testing data were generated with $\texttt{ppw} = 5$ using fourth-order Jacobi polynomials. Hence, we ensure that the mesh points are mostly non-overlapping for the datasets, likewise the Gauss-Lobatto nodes for each element. All simulations span frequencies up to 1,000 Hz with an average grid resolution of $\Delta x_{5\text{ppw}} = 0.069$ m and $\Delta x_{6\text{ppw}} = 0.057$ m when using $\texttt{ppw} = 5$ and $\texttt{ppw} = 6$, respectively. Testing data was generated with five source positions only.

The time step was $\Delta t = \text{CFL} \times \Delta x/c$ with the Courant-Friedrichs-Lewy constant set to $\text{CFL} = 1.0$ and $\text{CFL} = 0.2$ for frequency-independent and dependent impedance boundaries, respectively. The generated data sets were pruned in the temporal dimensions with $\texttt{ppw} \sim 2$, corresponding to a temporal resolution of $5e^{-4}$ s. Training the models on sparse temporal data results in overfitting, which we exploit for faster training and smaller data sets since interpolation in time is not useful for the applications of interest. The input function $u(x_i)$ for the branch net was uniformly sampled at the Nyquist limit $\texttt{ppw} = 2$ in the bounding box enclosing the geometry as depicted in the Figure B.10. This approach facilitates transfer learning but has the disadvantage of unnecessarily large input sizes for non-rectangular domains. The density of the source positions was determined by distributing the source positions with one-fifth wavelength for the training data and roughly one full wavelength for the validation data.

Before training, the spatial data has been normalized as a pre-processing step in the range $[-1, 1]$. The temporal dimension is normalized with the spatial normalization factor to ensure equal numerical resolutions in all dimensions of the temporal-spatial domain: e.g., if the spatial data is in the range $\xi \in [-2, 2]$ m and the temporal data is in the range $t = [0, 10]$ s, then the normalization factor is 2, and the temporal data would be normalized as $t_{\text{norm}} = [0, 5]$ s. To summarize, the data set has been constructed as

$$\begin{aligned} \mathcal{D}_j = \{\mathbf{u}_j, \xi_i\}_{i=1}^{N_{\text{full}}}, \quad \text{for} \quad j = 1, 2 \dots M_{\text{full}}, \quad \text{where} \\ \mathbf{u}_j = \{u_{j,i}\}_{i=1}^{m}, \qquad \xi_i = \{x_i, y_i, z_i, t_i\}, \end{aligned} \tag{3}$$

N_{full} is the number of spatiotemporal samples and M_{full} is the number of (Gaussian) source

functions $\mathbf{u}_j$. $\mathbf{u}_j$ is sampled at m fixed sensor locations used as input to the branch net, and ξ are the spatiotemporal samples used as input to the trunk net.

For training, a single mini-batch for each iteration is compiled by randomly sampling N input sample functions $\{\mathbf{u}^{(i)}\}_{i=1}^{N}$ for the branch net and randomly sampling Q coordinate pairs $\{\xi^{(i)} = (x_i, y_i, z_i, t_i)\}_{i=1}^{Q} \in \mathbb{R}^D$ for the trunk net for *each* input function. The details of network architecture and mini-batches are provided in Appendix B.1.

4.1.2. DeepONet

The DeepONet architecture used in this work is depicted in Figure B.7. In the literature, the DeepONet models have mostly been trained using Gaussian random fields (GRFs) as input to the branch net. However, this work uses the GIF from Equation 2 with $\sigma_0 = \frac{c}{\pi \cdot f_{\max}/2} = 0.22$ m spanning frequencies up to $f_{\text{phys}} = 1,000$ Hz and is used as a sound source input (initial condition) to the branch net. Using GIFs as ICs drastically reduces the number of samples needed for training compared to GRFs. Limiting the input space to Gaussian functions has no practical limitations in room acoustics since the room impulse response emitting from a GIF can be convolved with any band-limited signal to achieve the acoustical room signal for a fixed frequency range.

The input to the trunk network is the location ξ where the operator is evaluated and consists of the spatial and temporal coordinates x, y, z and t. To overcome the spectral bias [28, 29], the temporal and spatial inputs are passed through a positional encoding mapping as shown in Equation 4 to learn the high-frequency modes of the data, where the frequencies $\mathbf{f}_j = [500, 250, 167]$ Hz have been chosen relative to the fundamental frequency $f_0 = 1{,}000$ Hz, resulting in $2 \times 4 \times 3 = 24$ (`sine` and `cos` $\Rightarrow 2$, $x, y, z, t \Rightarrow 4$, expansion terms $\Rightarrow m = 3$) additional inputs to the trunk net.

$$\gamma(\mathbf{x}) = \left[\ldots, \cos\left(2\pi f_j \mathbf{x}\right), \sin\left(2\pi f_j \mathbf{x}\right), \ldots\right]^T, \quad \text{for } j = 0, \ldots, m-1. \tag{4}$$

The modified MLP architecture described in Appendix B.1.2 was used for the branch and trunk net. Self-adaptive weights were applied to all spatiotemporal locations optimized using a separate ADAM optimizer. Gradient clipping with an absolute value of 0.1 was needed to limit fluctuations that could sometimes make the optimizer jump to a drastically larger loss. The weights of the networks are initialized [30] as $w_i \sim \mathcal{U}\left(-\frac{\sqrt{6/n}}{k}, \frac{\sqrt{6/n}}{k}\right)$, where n denotes the number of input neurons to the i'th neuron and k was empirically chosen as $k = 30$ for all layers except for $k = 1$ used at the first layer. The first layer is initialized with weights such that the sine functions $\sin(w_0 \cdot \mathbf{W}\mathbf{x} + \mathbf{b})$ spans multiple periods, where the angular frequency $w_0 = 30$ was empirically found to give the best results.

4.1.3. Domain decomposition

When the frequency range is increased or, correspondingly, the domain size is increased, the accuracy of the deep neural network will decrease for a fixed network size. Increasing the network size in terms of layers and neurons should theoretically be sufficient to regain the required accuracy; however, this is often not the case. This is well-known in the literature and applies to, but is not limited to, both DeepONet and PINNs. Domain decomposition approaches, such as XPINNs [31] applied for PINNs, have been shown to overcome these

limitations, but with the expense of more neural networks to train. The general idea is to split the domain into (non-overlapping) partitions, each running separate neural networks and adding an additional loss term at the interface, imposing continuity conditions. In this work, training the DeepONet is purely data-driven, and a simpler approach has been taken. We divide the full domain $\Omega \in \mathbb{R}^3$ into four non-overlapping partitions $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$, where $\{x_i^{(k)}, y_i^{(k)}, z_i^{(k)}\}_{i=1}^{N_k} \in \Omega_k$, $k = 1 \dots 4$ and $N = \sum_{k=1}^{4} N_i$, $N_i \in \mathbb{Z}^+$. We then train four DeepONets $\mathcal{NN}_k$, each on the full source function space $\mathbf{u}$, but restrict the location where we evaluate the operator at one of the partitions Ω_i. The temporal samples are kept (could also be partitioned if needed), which gives us the data set $\mathcal{D}$ for training a DeepONet $\mathcal{NN}_k$ for a k'th partition

$$\mathcal{D}_j^{(k)} = \{\mathbf{u}_j, \xi_i^{(k)}\}_{i=1}^{N_k}, \quad \text{for} \quad j = 1, 2 \dots N, \quad \text{where} \\ \xi_i^{(k)} = \{x_i^{(k)}, y_i^{(k)}, z_i^{(k)}, t_i\}. \tag{5}$$

This work does not enforce continuity at the interfaces but could be done by calculating the mean of overlapping domains near the interfaces.

4.1.4. Self-adaptive weights

Weighting individual samples in the loss function can be advantageous for the DeepONet to perform better. Using point-wise weights, the loss function can be minimized w.r.t. the network parameters but maximized w.r.t. the point-wise loss weights. This approach, called self-adaptive weights, was originally introduced to improve the performance for PINNs [32] and later extended to DeepONet [33]. The self-adaptive weights are applied to all spatiotemporal samples and initialized to 1. To ensure stability in case some sample points are not converging, the weights have been clamped to take values between 0 and 1,000. A separate ADAM optimizer was used for updating the self-adaptive weights with a learning rate two orders of magnitude lower than the learning rate for the network parameters.

4.1.5. Transfer learning

Training DeepONet surrogate models for every geometry might get intractable for real-world usage due to the time and resources needed to train realistic 3D geometries. A more tractable strategy that could be applied for real-world problems is to pre-train DeepONets for geometries with certain traits (e.g., cubic rooms, L-shaped rooms, penta shapes, furnished rooms, etc.) and fine-tune the training on specific target room geometry by transferring the weight from a pre-trained DeepONet corresponding to the closest-matching geometry. We have made an investigation in 2D by performing transfer learning between rectangular, L-shape, and furnished geometries of varying sizes. First, the source models are trained using a network with two hidden layers of width 2,048 for both the branch and the trunk net using mini-batching of $N = 64$ and $Q = \{200, 600\}$. Then, the optimized network parameters are used to initialize the target model, a subset of the layers are frozen, and the new model is fine-tuned on data corresponding to the new geometry with $N = 64$ and $Q = \{200, 600\}$ on the full training set or a subset using only 60% of the Gaussian input functions (i.e., source positions). When freezing layers, the optimizer will skip updating the corresponding weights and biases for these. By sampling the Gaussian input function on an enclosing rectangle, the mapping from the target model's source positions to the source model's closest corresponding source positions can be done straightforwardly as shown in

the Figure B.10. It is also important to ensure that the spatial locations between the source and target model are as closely related as possible. For all the cases, the spatial alignment between source and receiver is done at the coordinate $[0, 0]$ m. The first hidden layer is frozen in the trunk net, leaving the second (non-linear) hidden layer and the linear output layer trainable. In contrast, only the linear output layer is trainable in the branch net. From the experiments, the trunk net learning the basis function is more important to fine-tuning the new geometry than the basis function coefficients learned by the branch net.

5. Acknowledgements

This research was conducted using computing resources and services at the Center for Computation and Visualization, Brown University, and at DTU Computing Center [23]. A big thanks go to Rômulo Silva for fruitful discussions. SG and GEK would like to acknowledge support by the MURI-AFOSR FA9550-20-1-0358 project.

References

[1] D. Botteldoorena, Finite-difference time-domain simulation of low-frequency room acoustic problems, Journal of the Acoustical Society of America 98 (6) (1995) 3302–3308.

[2] S. Bilbao, Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation, IEEE Transactions on Audio, Speech and Language Processing 21 (2013) 1524–1533.

[3] F. Pind, A. P. Engsig-Karup, C.-H. Jeong, J. S. Hesthaven, M. S. Mejling, J. Strømann-Andersen, Time domain room acoustic simulations using the spectral element method, Journal of the Acoustical Society of America 145 (6) (2019) 3299–3310.

[4] A. Melander, E. Strøm, F. Pind, A. Engsig-Karup, C.-H. Jeong, T. Warburton, N. Chalmers, J. S. Hesthaven, Massive parallel nodal discontinuous galerkin finite element method simulator for room acoustics, Infoscience EPFL scientific publications (preprint) (2020).

[5] S. Kirkup, The Boundary Element Method in Acoustics, Vol. 8, 2007.

[6] M. Hornikx, R. Waxler, J. Forssén, The extended fourier pseudospectral time-domain method for atmospheric sound propagation, Journal of the Acoustical Society of America 128 (4) (2010) 1632–1646.

[7] S. Greenwold, Spatial computing, Master's thesis, Massachusetts Institute of Technology (2003).

[8] N. Borrel-Jensen, A. Engsig-Karup, C.-H. Jeong, Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries, Journal of the Acoustical Society of America Express Letters 1 (12) (2021).

[9] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Transactions on Neural Networks 6 (4) (1995) 911–917.

[10] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature Machine Intelligence 3 (3) (2021) 218–229.

[11] S. Goswami, M. Yin, Y. Yu, G. E. Karniadakis, A physics-informed variational deeponet for predicting crack path in quasi-brittle materials, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114587.

[12] V. Kumar, S. Goswami, D. J. Smith, G. E. Karniadakis, Real-time prediction of multiple output states in diesel engines using a deep neural operator framework, arXiv preprint arXiv:2304.00567 (2023).

[13] V. Oommen, K. Shukla, S. Goswami, R. Dingreville, G. E. Karniadakis, Learning two-phase microstructure evolution using neural operators and autoencoder architectures, npj Computational Materials 8 (1) (2022) 190.

[14] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, G. E. Karniadakis, Operator learning for predicting multiscale bubble growth dynamics, Journal of Chemical Physics 154 (10) (2021) 104118.

[15] S. Goswami, D. S. Li, B. V. Rego, M. Latorre, J. D. Humphrey, G. E. Karniadakis, Neural operator learning of heterogeneous mechanobiological insults contributing to aortic aneurysms, Journal of the Royal Society Interface 19 (193) (2022) 20220410.

[16] K. Shukla, V. Oommen, A. Peyvan, M. Penwarden, L. Bravo, A. Ghoshal, R. M. Kirby, G. E. Karniadakis, Deep neural operators can serve as accurate surrogates for shape optimization: a case study for airfoils, arXiv preprint arXiv:2302.00807 (2023).

[17] S. Goswami, K. Kontolati, M. D. Shields, G. E. Karniadakis, Deep transfer operator learning for partial differential equations under conditional shift, Nature Machine Intelligence 4 (12) (2022) 1155–1164.

[18] H. Kuttruff, Room Acoustics, 6th Edition, CRC Press, 2016.

[19] Iso 3382-1 (2023).
URL https://www.iso.org/standard/40979.html

[20] N. Raghuvanshi, J. Snyder, Parametric wave field coding for precomputed sound propagation, ACM Transactions on Graphics 33 (4) (2014).

[21] J. Sandvad, Dynamic aspects of auditory virtual environments, in: Audio Engineering Society Convention 100, 1996.

[22] N. Borrel-Jensen, A. P. Engsig-Karup, C.-H. Jeong, A sensitivity analysis on the effect of hyperparameters in deep neural operators applied to sound propagation, Forum Acusticum (9 2023).

[23] DTU Computing Center, DTU Computing Center resources (2022).

[24] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2022).

[25] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, M. van Zee, Flax: A neural network library and ecosystem for JAX (2023).
URL http://github.com/google/flax

[26] Y. Miki, Acoustical properties of porous materials-modifications of delany-bazley models-, Journal of the Acoustical Society of Japan (E) 11 (1) (1990) 19–24.

[27] B. Gustavsen, A. Semlyen, Rational approximation of frequency domain responses by vector fitting, IEEE Transactions on Power Delivery 14 (3) (1999) 1052–1061.

[28] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, arXiv preprint arXiv:1806.08734 (6 2018).

[29] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, S. Kritchman, Frequency bias in neural networks for input of non-uniform density, arXiv preprint arXiv:2003.04560 (3 2020).

[30] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, Advances in Neural Information Processing Systems (2020).

[31] A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Communications in Computational Physics 28 (5) (2020) 2002–2041.

[32] L. D. McClenny, U. M. Braga-Neto, Self-adaptive physics-informed neural networks, Journal of Computational Physics 474 (2023) 111722.

[33] K. Kontolati, S. Goswami, M. D. Shields, G. E. Karniadakis, On the influence of overparameterization in manifold based surrogates and deep neural operators, Journal of Computational Physics (2023) 112008.

[34] J. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, Springer, 2015.

[35] H. S. Llopis, A. P. Engsig-Karup, C.-H. Jeong, F. Pind, J. S. Hesthaven, Reduced basis methods for numerical room acoustic simulations with parametrized boundaries, Journal of the Acoustical Society of America 152 (2022) 851–865.

[36] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[37] N. Raghuvanshi, Dynamic Portal Occlusion for Precomputed Interactive Sound Propagation, arXiv preprint arXiv:2107.11548 (Jul. 2021).

[38] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020).

[39] T. Tripura, S. Chakraborty, Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115783.

[40] Q. Cao, S. Goswami, G. E. Karniadakis, Lno: Laplace neural operator for solving differential equations, arXiv preprint arXiv:2303.10528 (2023).

[41] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM Journal on Scientific Computing 43 (5) (2021) A3055–A3081.

[42] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, Sci. Adv 7 (2021) 8605–8634.

[43] S. Wang, H. Wang, P. Perdikaris, Improved Architectures and Training Algorithms for Deep Operator Networks, Journal of Scientific Computing 92 (2) (2022) 35.

[44] R. Troian, D. Dragna, C. Bailly, M. A. Galland, Broadband liner impedance eduction for multimodal acoustic propagation in the presence of a mean flow, Journal of Sound and Vibration 392 (2017) 200–216.

Appendix A. Parameterized PDEs in acoustics

The challenge of utilizing parametric PDEs has motivated increased research. Reduced order methods [34, 35] aim to reduce the degrees of freedom; however, despite achieving orders of magnitude in accelerations for many applications, these techniques still cannot meet the runtime requirement for real-time experiences for sound propagation in realistic 3D schenes. Recently, the possibility of generating surrogate models with little data was demonstrated using physics-informed neural networks [36] and applied for acoustics problems in [8]. Previous attempts to overcome the storage requirements of the IR include work for lossy compression [20]. Lately, a novel portal search method has been proposed as a drop-in solution to pre-computed IRs to adapt to flexible scenes, e.g., when doors and windows are opened and closed [37].

Appendix B. Methods

Appendix B.1. Neural operators

Let $\Omega \subset \mathbb{R}^D$ be a bounded open set and $\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^{d_x})$ and $\mathcal{Y} = \mathcal{Y}(\Omega; \mathbb{R}^{d_y})$ two separable Banach spaces. Furthermore, assume that $\mathcal{G} : \mathcal{U} \to \mathcal{Y}$ is a non-linear map arising from the solution of a time-dependent PDE. The objective is to approximate the nonlinear operator via the following parametric mapping

$$\mathcal{G} : \mathcal{U} \times \Theta \to \mathcal{Y} \quad \text{or,} \quad \mathcal{G}_\theta : \mathcal{U} \to \mathcal{Y}, \ \theta \in \Theta \tag{B.1}$$

where Θ is a finite-dimensional parameter space. The optimal parameters θ^* are learned via the training of a neural operator with backpropagation based on a dataset $\{\mathbf{u}_j, \mathbf{y}_j\}_{j=1}^N$ generated on a discretized domain $\Omega_m = \{x_1, \ldots, x_m\} \subset \Omega$ where $\{x_j\}_{j=1}^m$ represent the sensor locations, thus $\mathbf{u}_{j|\Omega_m} \in \mathbb{R}^{D_x}$ and $\mathbf{y}_{j|\Omega_m} \in \mathbb{R}^{D_y}$ where $D_x = d_x \times m$ and $D_y = d_y \times m$.

Appendix B.1.1. The deep operator network (DeepONet)

DeepONet [10] aims to learn operators between infinite-dimensional Banach spaces. Learning is performed in a general setting in the sense that the sensor locations $\{x_i\}_{i=1}^m$ at which the input functions are evaluated need not be equispaced; however, they need to be consistent across all input function evaluations. Instead of blindly concatenating the input data (input functions $[\mathbf{u}(x_1), \mathbf{u}(x_2), \ldots, \mathbf{u}(x_m)]^T$ and locations ζ) as one input, i.e., $[\mathbf{u}(x_1), \mathbf{u}(x_2), \ldots, \mathbf{u}(x_m), \zeta]^T$, DeepONet employs two subnetworks and treats the two inputs equally. Thus, DeepONet can be applied for high-dimensional problems where the dimension of $\mathbf{u}(u_i)$ and ζ no longer match since the latter is a vector of d components in total. A trunk network $\mathbf{f}(\cdot)$, takes as input ζ and outputs $[tr_1, tr_2, \ldots, tr_p]^T \in \mathbb{R}^p$ while a second network, the branch net $\mathbf{g}(\cdot)$, takes as input $[\mathbf{u}(x_1), \mathbf{u}(x_2), \ldots, \mathbf{u}(x_m)]^T$ and outputs $[b_1, b_2, \ldots, b_p]^T \in \mathbb{R}^p$. Both subnetwork outputs are merged through a dot product to generate the quantity of interest. A bias $b_0 \in \mathbb{R}$ is added in the last stage to increase expressivity, i.e., $\mathcal{G}(\mathbf{u})(\zeta) \approx \sum_{i=k}^p b_k t_k + b_0$. The generalized universal approximation theorem for operators, inspired by the original theorem introduced by [9], is presented below. The generalized theorem essentially replaces shallow networks used for the branch and trunk net in the original work with deep neural networks to gain expressivity. An overview of the architecture used in this work is depicted in Figure B.7.

Theorem 1 (Generalized Universal Approximation Theorem for Operators.)**.** *Suppose that X is a Banach space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$. Assume that: $\mathcal{G} : V \to C(K_2)$ is a nonlinear continuous operator. Then, for any $\epsilon > 0$, there exist positive integers m, p, continuous vector functions $\mathbf{g} : \mathbb{R}^m \to \mathbb{R}^p$, $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^p$, and $x_1, x_2, \ldots, x_m \in K_1$ such that*

$$\left| \mathcal{G}(\mathbf{u})(\zeta) - \langle \underbrace{\mathbf{g}(\mathbf{u}(x_1), \mathbf{u}(x_2), \ldots, \mathbf{x}(x_m))}_{branch}, \underbrace{\mathbf{f}(\zeta)}_{trunk} \rangle \right| < \epsilon$$

holds for all $\mathbf{u} \in V$ and $\zeta \in K_2$, where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathbb{R}^p$. For the two functions $\mathbf{g}, \mathbf{f}$ classical deep neural network models and architectures can be chosen that satisfy the universal approximation theorems of functions, such as fully-connected networks or convolutional neural networks.

The method accurately learns the mapping from an input space of functions into a space of output functions, thereby generalizing the solution for a parametrized PDE. DeepONet provides a simple architecture that is fast to train, utilizing data from high-fidelity simulations describing sound propagation, and allows for continuous target outputs predicting source/receiver pairs in a grid-less domain almost instantly.

The Fourier neural operator [38], Wavelet neural operator [39], and the Laplace neural operator [40] are a separate class of neural operator where the solution operator is expressed as an integral operator of Green's function that is parameterized in the Fourier, Wavelet, and Laplace space, respectively. All these versions are different realizations of DeepONet if appropriate changes are imposed on its architecture. Approximating operators is a paradigm shift from current and established machine learning techniques focusing on function approximation to the solution of the PDEs.

Appendix B.1.2. DeepONet architecture

The DeepONet framework allows many network architectures, such as feed-forward neural networks (FNN), multi-layer perception (MLP), recurrent neural networks (RNN), convolutional neural networks (CNN), graph neural networks (GNN), and convolutional graph neural networks (CGNN). In this work, we have used a modification to the MLP (mod-MLP) for both the branch and trunk net originally proposed in [41] for PINNs and in [42] for DeepONets shown to outperform the conventional FNNs. First, let us define a standard FNN consisting of an input layer $\mathbf{x}$, n hidden layers, and an output layer. The mapping from an input $\mathbf{x}$ to an output $\mathbf{y}$ is defined as

$$\begin{aligned} \mathbf{y} &= (f_0 \circ f_1 \circ \ldots \circ f_n)(\mathbf{x}), \\ f_i(\mathbf{x}) &= \sigma_i(\mathbf{W}^i \mathbf{x} + \mathbf{b}^i). \end{aligned} \tag{B.2}$$

$\sigma_i(\mathbf{x})$ is a non-linear activation function (except for a linear activation in the last layer), where $\mathbf{W}^i$ and $\mathbf{b}^i$ are the weight and bias parameters to learn. An MLP is a special case of an FNN, where every layer is fully connected, and the number of nodes in each layer is the same. The key extension is the introduction of two encoder networks encoding the input variables to a higher-dimensional feature space. The networks consisting of a single layer are shared between all layers, and a pointwise multiplication operation is performed to update the hidden layers. Let the two shallow encoder networks with width size equal to the hidden

layers be denoted $u(\mathbf{x})$ and $v(\mathbf{x})$ and defined as a simple perceptron

$$\begin{aligned} u(\mathbf{x}) &= \sigma(\mathbf{W}_u \mathbf{x} + \mathbf{b}_u), \\ v(\mathbf{x}) &= \sigma(\mathbf{W}_v \mathbf{x} + \mathbf{b}_v), \end{aligned} \tag{B.3}$$

then the mod-MLP is defined as

$$\begin{aligned} \mathbf{y} = &((1 - f_0) \odot u + f_0 \odot v) \circ \\ &((1 - f_1) \odot u + f_1 \odot v) \circ \\ &\vdots \\ &((1 - f_n) \odot u + f_n \odot v)(\mathbf{x}), \end{aligned} \tag{B.4}$$

where $\odot$ denotes elementwise multiplication, $\circ$ is the function composition operator, and $\mathbf{W}_{\{u,v\}}$ and $\mathbf{b}_{\{u,v\}}$ are the weights and biases for the two encoder networks. The architecture is depicted in Figure B.8, where the encoder networks are applied separately for the branch and trunk net. This implementation differs from the implementation in [43], where only two encoder networks are shared between the branch and trunk layer. The motivation behind the mod-MLP is to better propagate information stably through the network since the trainability of the DeepONet depends on merging the branch and trunk net in terms of their inner product only in the last layer. Hence, if the input signals are not properly propagated through the network, this may lead to ineffective training and poor model performance.

Appendix B.1.3. DeepONet setup

Five hidden layers with 2,048 neurons each were used for the branch and trunk net in 3D; two hidden layers with 2,048 neurons each were used for the branch and trunk net in 2D. The ADAM optimizer and the mean-squared error for calculating the losses were used with a learning rate of $1e-3$ and exponential decay of 0.90 per 2,000 iterations for all experiments. Self-adaptive weights were applied to all spatiotemporal locations using a separate ADAM optimizer with a learning rate two orders of magnitude smaller than the learning rate of the optimizer used for the network parameters. All experiments used mini-batches of $N = 64$, $Q = 1{,}000$, except for the dome, where mini-batches of $N = 96$, $Q = 1{,}500$ were used. For the transfer learning in 2D, $N = 64$ and $Q = \{200, 600\}$ were used for training the reference and target models. Note, that the data set batch dimensions $\mathbf{u}$, ξ, $G(\mathbf{u})(\xi)$ are $(N \times Q, m)$, $(N \times Q, D)$, $(N \times Q, 1)$, respectively.

Appendix B.2. Impedance boundaries

We consider impedance boundaries and denote the boundary domain as Γ. We will omit the source position x_0 in the following. For frequency-independent impedance boundaries, the acoustic properties of a wall can be described by its surface impedance $Z_s = \frac{p}{v_n}$ [18] where v_n is the normal component of the velocity at the same location on the wall surface. Combining the surface impedance with the pressure term $\frac{\partial p}{\partial \mathbf{n}} = -\rho_0 \frac{\partial v_n}{\partial t}$ of the linear coupled wave equation yields

$$\frac{\partial p}{\partial t} = -c\xi_{\text{imp}} \frac{\partial p}{\partial \mathbf{n}}, \quad \text{for} \quad \Omega \quad \text{and} \quad t \geq 0, \tag{B.5}$$

where $\xi_{\text{imp}} = Z_s/(\rho_0 c)$ is the normalized surface impedance and ρ_0 denotes the air density (kg/m^3). Note that perfectly reflecting boundaries can be obtained by letting $\xi_{\text{imp}} \to \infty$ be the Neumann boundary formulation.

	Source function		Location		
	Count	Sensors	Time steps	Mesh points	Total per source
T - Cubic	2,826	28	101	57,124	5.8M
V - Cubic	100	28	101	17,643	1.8M
T - L-shape	5,165	3,888	101	93,675	9.5M
V - L-shape	180	3,888	101	51,201	5.2M
T - Furn.	4,799	3,888	101	123,994	12.5M
V - Furn.	203	3,888	101	74,819	7.6M
T - Dome	1,849	19,602	101	213,130	21.5M
V - Dome	94	19,602	101	165,025	16.7M*
T - Dome 1/4	1,849	19,602	101	51,665	5.2M
V - Dome 1/4	94	19,602	101	40,240	4.1M*

Table B.1: Data sizes for the four geometries. The data has been saved in 16-bit floating point precision. The dome 1/4 arises from being spatially partitioned into four partitions, subsequently evaluated at one partition only. 'T' denotes training data, 'V' denotes validation data. *Note that the mesh point ratio between training and validation data differs for the dome compared to the other geometries. This is caused by the meshing algorithm forcing finer resolutions in regions near the sphere to capture the complex geometry.

For frequency-dependent impedance boundaries, the wall impedance can be written as a rational function in terms of the admittance $Y = 1/Z_s$ and rewritten by using partial fraction decomposition in the last equation [44]

$$\begin{aligned} Y(\omega) &= \frac{a_0 + \ldots + a_N(-i\omega)^N}{1 + \ldots + b_N(-i\omega)^N} \\ &= Y_\infty + \sum_{k=0}^{Q-1} \frac{A_k}{\lambda_k - i\omega} + \sum_{k=0}^{S-1} \left(\frac{B_k + iC_k}{\alpha_k + i\beta_k - i\omega} + \frac{B_k - iC_k}{\alpha_k - i\beta_k - i\omega} \right), \end{aligned} \tag{B.6}$$

where a, b are real coefficients, $i = \sqrt{-1}$ being the complex number, Q is the number of real poles λ_k, S is the number of complex conjugate pole pairs $\alpha_k \pm j\beta_k$, and Y_∞, A_k, B_k and C_k are numerical coefficients. Since we are concerned with the (time-domain) wave equation, the inverse Fourier transform is applied to the admittance and the partial fraction decomposition term in Equation B.6. Combining these gives [44]

$$v_n(t) = Y_\infty p(t) + \sum_{k=0}^{Q-1} A_k \phi_k(t) + \sum_{k=0}^{S-1} 2 \left[B_k \psi_k^{(0)}(t) + C_k \psi_k^{(1)}(t) \right]. \tag{B.7}$$

The functions ϕ_k, $\psi_k^{(0)}$, and $\psi_k^{(1)}$ are the so-called accumulators determined by the following set of ordinary differential equations (ODEs) referred to as auxiliary differential equations (ADEs)

$$\frac{d\phi_k}{dt} + \lambda_k \phi_k = p, \qquad \frac{d\psi_k^{(0)}}{dt} + \alpha_k \psi_k^{(0)} + \beta_k \psi_k^{(1)} = p, \qquad \frac{d\psi_k^{(1)}}{dt} + \alpha_k \psi_k^{(1)} - \beta_k \psi_k^{(0)} = 0. \tag{B.8}$$

The boundary conditions can then be formulated by inserting the velocity v_n calculated in Equation B.7 into the pressure term of the linear coupled wave equation $\frac{\partial p}{\partial \mathbf{n}} = -\rho_0 \frac{\partial v_n}{\partial t}$.

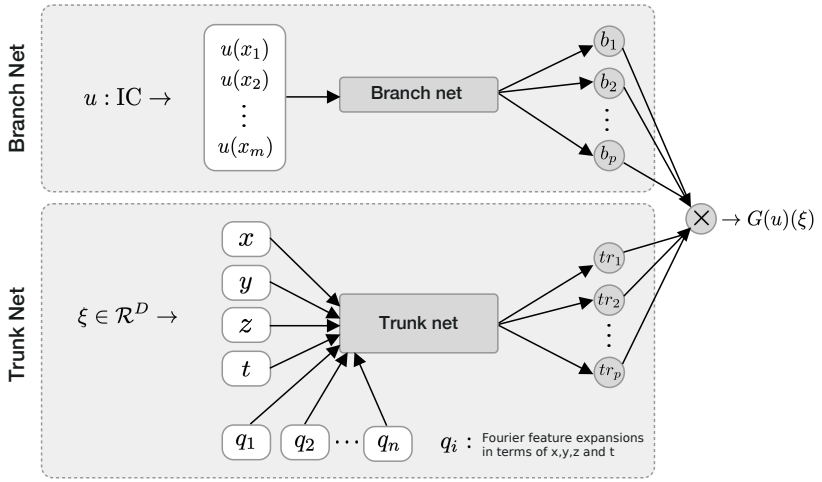


Figure B.7: DeepONet architecture with parameterized source position for predicting the impulse response for a source/receiver pair over time for a 3D domain. The branch net is taking as input a Gaussian source function $\mathbf{u}$ determining the source position, sampled at fixed sensor locations. The spatial coordinates x, y, z, and temporal coordinate t are denoted by ξ and are used as input to the trunk net mapping into the output domain of the operator.

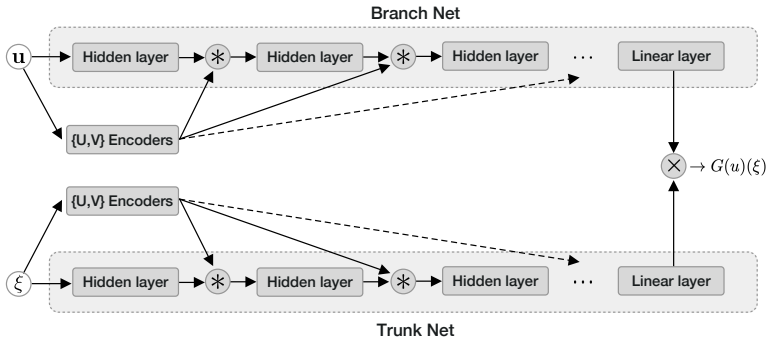


Figure B.8: The modified MLP architecture applied for the DeepONet. Two encoders u and v implemented as single-layer neural networks are applied for each MLP, embedding the inputs into a latent space with the size of the layer width of the MLP. The embedded features are then inserted into each hidden layer illustrated by '*' performing the operation $(1 - f_i) \odot u + f_i \odot v$.

Domain	s_1 RMSE	s_2 RMSE	s_3 RMSE	s_4 RMSE	s_5 RMSE	Mean RMSE
Cubic	0.03 Pa	0.03 Pa	0.02 Pa	0.04 Pa	0.03 Pa	0.03 Pa
L-shape	0.06 Pa	0.04 Pa	0.05 Pa	0.04 Pa	0.04 Pa	0.05 Pa
Furnished	0.09 Pa	0.09 Pa	0.09 Pa	0.08 Pa	0.08 Pa	0.09 Pa
Dome	0.08 Pa	0.05 Pa	0.08 Pa	0.10 Pa	0.10 Pa	0.08 Pa
Dome 1/4	0.03 Pa	0.02 Pa	0.04 Pa	0.04 Pa	0.04 Pa	0.03 Pa

Table B.2: Impulse receiver errors for source/receiver pairs s_i given in the text. The root mean square error (RMSE) is used to access the errors, defined as RMSE $= \sqrt{\frac{\sum_{n=1}^{N}(p_{\mathrm{ref}_i} - p_{\mathrm{pred}_i})^2}{N}}$.

		Layer	inputs	outputs	param.	size
Cubic	Branch net	U,V	1,728	2,048	2×3.5M	2×14 MB
		Hidden layer (in)	1,728	2,048	3.5M	14 MB
		Hidden layers	2,048	2,048	4×4.2M	4×17 MB
		Output layer	2,048	100	204k	820 KB
		Total	-	-	27.6M	111 MB
Furnished	Branch net	U,V	3,888	2,048	2×8M	2×32 MB
		Hidden layer (in)	3,888	2,048	8M	32 MB
		Hidden layers	2,048	2,048	4×4M	4×17 MB
		Output layer	2,048	100	204k	820 KB
		Total	-	-	40.9M	164 MB
L-shape	Branch net	U,V	3,888	2,048	2×8M	2×32 MB
		Hidden layer (in)	3,888	2,048	8M	32 MB
		Hidden layers	2,048	2,048	4×4M	4×17 MB
		Output layer	2,048	100	204k	820 KB
		Total	-	-	40.9M	164 MB
Dome	Branch net	U,V	19,602	2,048	2×40.1M	2×161MB
		Hidden layer (in)	19,602	2,048	40.1M	161 MB
		Hidden layers	2,048	2,048	4×4.2M	4×17 MB
		Output layer	2,048	100	204k	820 KB
		Total	-	-	137M	549 MB
[all]	Trunk net	U,V	28	2,048	2×59k	2×238k
		Hidden layer (in)	28	2,048	59k	238k
		Hidden layers	2,048	2,048	4×4M	4×17 MB
		Output layer	2,048	100	204k	820 KB
		Total	-	-	17M	69 MB

Table B.3: Branch and trunk network parameters for the cubic, L-shape, furnished, and dome geometries. The input function to the branch net has been uniformly sampled on the enclosed bounding box for the geometries; why the input size is the same for the L-shape and furnished rooms both having outer dimension 3m × 3m × 2m. The trunk net is the same for all geometries.

	Timings			Data size
	per iter	loading	back-prop	per iter
2D Furn.	32.7 ms	1.3 ms/3.8%	31.4 ms/96.2%	0.024 MB
3D Furn.	2.1 s	1.6 s/73%	564 ms/27%	1.5 GB
Factor 3D/2D	64×	1230×	18×	62,500×

Table B.4: Training time divided into data loading and weight/bias updates through forward/back-propagation. The timings are given per iteration step for the furnished room in 2D and 3D. The 2D network has two layers of width 2,048 for the BN and TN with batch size $64 \times 200 = 12{,}800$, and all data fits into memory for fast access and efficient sampling. The 3D network has five layers of width 2,048 for the BN and TN with batch size $64 \times 1{,}000 = 64{,}000$. The data is stored in HDF5 format in separate files for each source position. Therefore, the source position can be sampled randomly by loading a subset of the HDF5 files. In contrast, the temporal/spatial data cannot efficiently be accessed randomly on disk. Therefore all data for each file are loaded in memory, taking up $64 \times 101 \times 123{,}994$ 16-bit samples (source sample × temporal dim. × spatial dim).

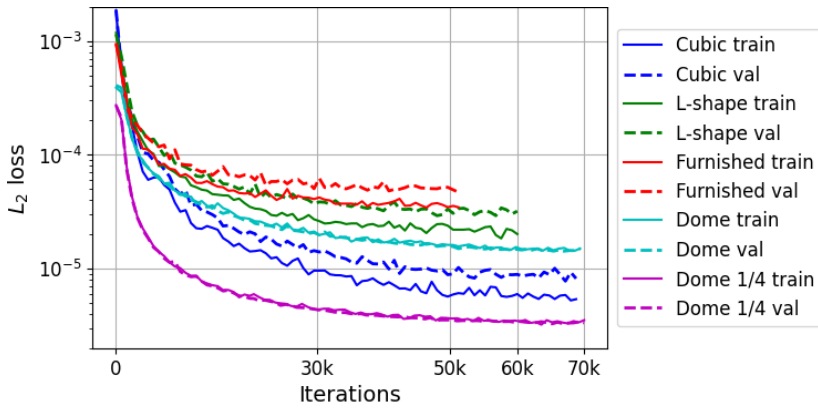


Figure B.9: Convergence plot showing the training and validation loss for the cubic, L-shape, furnished, and dome geometries.

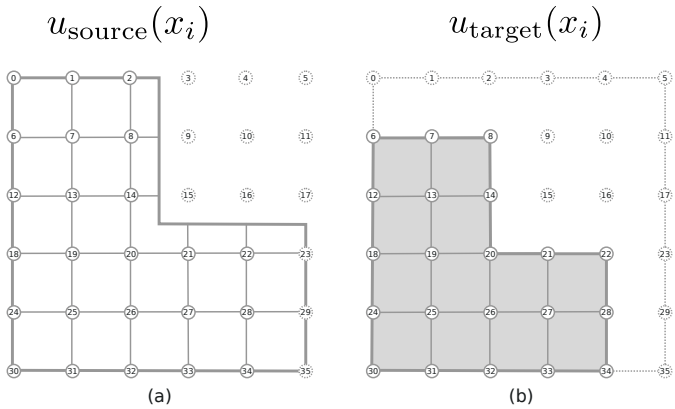


Figure B.10: The input function u is uniformly sampled at N fixed locations x_i for $i = 1, 2, \ldots, N$ on a bounding box enclosing the geometry. The dots represent the discretization x_i. Sampling the input function this way facilitates transfer learning, where similar pressure values between domains are kept fixed. (a) Initial condition grid, flattened for input to the branch net as $\mathbf{u} = [x_0, x_1, \ldots, x_{35}]$, where the ghost nodes are set to zero pressures $[x_i = 0 | i \in \{3, 4, 5, 9, 10, 11, 15, 16, 17\}]$, *b*) Modified initial condition grid preserving the ordering by keeping the source grid and setting the new ghost nodes to zero $[x_i = 0 | i \in \{0, \ldots, 5, 9, \ldots, 11, 15, \ldots, 17, 23, 29, 35\}]$.

Paper C

forum acusticum 2023

A SENSITIVITY ANALYSIS ON THE EFFECT OF HYPERPARAMETERS IN DEEP NEURAL OPERATORS APPLIED TO SOUND PROPAGATION

Nikolas Borrel-Jensen[1*] **Allan P. Engsig-Karup**[2]
Cheol-Ho Jeong[1]
[1] Technical University of Denmark, Dep. of Electrical and Photonics Engineering, DK
[2] Technical University of Denmark, Dep. of Applied Mathematics and Computer Science, DK

ABSTRACT

Deep neural operators have seen much attention in the scientific machine learning community over the last couple of years due to their capability of efficiently learning the nonlinear operators mapping from input function spaces to output function spaces showing good generalization properties. This work will show how to set up a performant DeepONet architecture in acoustics for predicting 2-D sound fields with parameterized moving sources for real-time applications. A sensitivity analysis is carried out with a focus on the choice of network architectures, activation functions, Fourier feature expansions, and data fidelity to gain insight into how to tune these models. Specifically, a default feed-forward neural network (FNN), a modified FNN, and a convolutional neural network will be compared. This work will de-mystify the DeepONet and provide helpful knowledge from an acoustical point of view.

Keywords: *neural operators, sensitivity analysis, virtual acoustics, DeepONet*

1. INTRODUCTION

Deep learning has seen rapid development over the last 20 years, with a many-fold of applications such as image classification and computer vision, speech recognition, language translation, autonomous driving, bioinformatics, chatbots, and more. More recently, scientific machine-learning (SciML) methods have been proposed in the context of deep learning. Physics-informed neural networks were introduced in 2017 [1] inspired by [2] and have already seen many applications [3, 4]. Typical for most of these techniques is that they are based on the assumption of function approximations of neural networks [5]. In 2019, the DeepONet was introduced [6], extending a theorem on the universal operator approximation for a single-layer neural network [7] to hold for deep neural networks. Moreover, the original architecture was improved to exhibit small generalization errors. Unlike function regression, operator regression aims to learn the mapping from one function space (inputs) to another function space (output), where the learned operator can be evaluated at arbitrary (continuous) locations. Another work on operator learning is the Fourier Neural Operator (FNO) introduced in 2021 [8], based on parameterizing an integral kernel directly in the Fourier space.

In this work, we will apply the DeepONet to approximate the wave equation operator for frequency-independent boundary conditions and parameterized source positions similar to [4] in 2-D for virtual acoustics applications such as computer games, AR/VR, and metaverses. The impulse response (IR) has traditionally been calculated using numerical methods. However, when handling moving sources, this approach gets challenging both from a computational and storage point of view since the IRs have to be re-calculated offline for each source position and the IRs stored for each source/receiver pair. This gets intractable when approaching the full frequency range.

It is vital to apply various techniques and fine-tune the hyperparameters for the DeepONet to exhibit fast

*__Corresponding author__: nibor@dtu.dk.

convergence with low generalization errors. This work aims to perform a sensitivity w.r.t. the network architecture, data resolution, batch size, activation functions, and Fourier feature expansion techniques to better understand the workings of the DeepONet when applied in an acoustical context.

2. METHODS

2.1 Deep operator network (DeepONet)

DeepONet [6] is a general deep learning framework for approximating continuous operators contrary to continuous functions. The underlying theory stems from the universal operator approximation theorem [7], stating that a neural network (NN) with a single hidden layer of infinite width can approximate any nonlinear continuous functional or operator. Let G be the operator we want to learn using NNs, defined as $G : u \mapsto G(u)$, where u is the input function to G and $G(u)$ is the output function. For any point, y in the domain of $G(u)$, $G(u)(y) \in \mathbb{R}$ is producing a real number. Translating this into a NN setting, the network takes two inputs, u and y, and outputs $G(u)(y)$. The input function is discretized by evaluating u at a finite number of points $\{x_i\}$ called 'sensors.' The approximation theorem by Chen and Chen considers shallow networks and only guarantees small approximation errors but does not consider generalization and optimization errors. In [6], the authors extended the original theorem by proposing deep neural networks instead of shallow networks and proved that the network is also universal approximators for operators. The proposed deep operator network, DeepONet, achieves small total errors, including approximation, optimization, and generalization errors. The DeepONet architecture consists of two subnetworks, the 'branch net' for the input functions and the 'trunk net' for the locations to evaluate the output function $G(u)$. The trunk network takes y as input and outputs $[T_1, T_2, \ldots, T_p] \in \mathbb{R}^p$; the branch network takes $[u(x_1), u(x_2), \ldots, u(x_m)]^T$ at fixed sensors $\{x_1, x_2, \ldots, x_m\}$ and outputs a scalar $B_k \in \mathbb{R}$ for $k = 1, 2, \ldots, p$. By merging the trunk and branch in terms of their inner product, we get

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{B_k\left(u(x_1), u(x_2), \ldots, u(x_m)\right)}_{\text{branch}} \underbrace{T_k(y)}_{\text{trunk}} + b_0, \tag{1}$$

where b_0 is a trainable bias.

2.2 Network architectures

The DeepONet framework allows many network architectures, such as feed-forward neural networks (FNN), recurrent neural networks (RNN), convolutional neural networks (CNN), graph neural networks (GNN), and convolutional graph neural networks (CGNN). Since the inputs to the trunk net are the (continuous) locations on which the operator's output is evaluated, the dimensionality is typically low. In contrast, the input to the branch net is a function sampled at m sensor locations and is typically high-dimensional. A common choice for the trunk network is to use an FNN, whereas FNNs and CNNs have been the most common architectures for the branch net. The latter choice is because of the high-dimensional nature of the input functions, making the CNNs particularly useful in many applications due to their ability to map higher-dimensional spaces to lower-dimensional spaces by extracting the essential features.

In this work, we compare the performance between using a CNN and an FNN network for the branch net of the DeepONet, while keeping the trunk net FNN architecture fixed.

2.2.1 Feed-Forward Neural Network

An FNN consists of an input layer $\mathbf{x}$, n hidden layers, and an output layer and maps an input $\mathbf{x}$ to an output $\mathbf{y}$ as

$$\mathbf{y} = (f_0 \circ f_1 \circ \ldots \circ f_n)(\mathbf{x}), \tag{2a}$$

$$f_i(\mathbf{x}) = \sigma_i(\mathbf{W}^i\mathbf{x} + \mathbf{b}^i), \tag{2b}$$

where $\sigma_i(\mathbf{x})$ is a non-linear activation function, except for the last layer, where we are applying the identity mapping $\sigma_n(\mathbf{x}) = \mathbf{x}$. The weights $\mathbf{W}^i$ and biases $\mathbf{b}^i$ are the parameters to learn. A multilayer perceptron (MLP) is a special case of an FNN, where every layer is fully connected, and the number of nodes in each layer is the same. In this work, we have used the gradient descent optimizer ADAM.

A modification to the MLP (mod-MLP) was proposed in [9] and has been shown to outperform the conventional FNNs also in conjunction with DeepONet [10]. The key extension is the introduction of two encoder networks encoding the input variables to a higher-dimensional feature space. The networks consisting of a single layer are shared between all layers, and a pointwise multiplication operation is performed to update the hidden layers. Let the two transformer networks be denoted $u(\mathbf{x})$ and $v(\mathbf{x})$ and de-

fined as a simple perceptron

$$u(\mathbf{x}) = \sigma(\mathbf{W}_u\mathbf{x} + \mathbf{b}_u), \tag{3a}$$
$$v(\mathbf{x}) = \sigma(\mathbf{W}_v\mathbf{x} + \mathbf{b}_v), \tag{3b}$$

then the mod-MLP is defined as

$$\begin{aligned}\mathbf{y} = &((1 - f_0) \odot u + f_0 \odot v) \circ \\ &((1 - f_1) \odot u + f_1 \odot v) \circ \\ &\vdots \\ &((1 - f_n) \odot u + f_n \odot v)(\mathbf{x}),\end{aligned} \tag{4}$$

where $\odot$ denotes elementwise multiplication, and $\mathbf{W}_{\{u,v\}}$ and $\mathbf{b}_{\{u,v\}}$ are the weights and biases for the two transformer networks.

2.2.2 *Convolutional Neural Network*

Convolutional neural networks (CNN) are special networks for processing data with a grid-like topology [11] and use convolution instead of matrix multiplication in one or more layers. In traditional MLPs, each output unit interacts with each input unit through weight parameters describing the interaction. In contrast, CNNs typically have sparse interactions by applying a convolution kernel (much) smaller than the input dimension. Moreover, the convolutional kernel is used for every input position, meaning the parameters are shared. Aside from reducing the storage requirement, it also causes the layer to have equivalence to translation.

Although the networks in our work are not particularly deep, we will use the ResNet architecture [12]. Several ResNet blocks assemble the ResNet. A ResNet block consists of two stacked CNNs with skip connections and batch normalization; one or more ResNet blocks comprise a group (all with the same output shape), and one or more groups connect the final ResNet. Downsampling takes place in the first block of each group by increasing the strides, and the output channel dimension is increased, forcing the CNN to capture essential features in separate channels. We will use the notation ResNet-{gr1,gr2,gr3,...}, where the element counts inside the square brackets denote the number of groups, and the values denote the number of blocks inside the group indexed by its position. The hidden channel layers are denoted {ch1,ch2,ch3,ch4,...}, where each element index corresponds to the ResNet group with the same index. E.g., when we refer to a ResNet-{2,2,2,2} with hidden channel layers {8,16,32,64}, it denotes a ResNet having 4 groups of 2 blocks each with 8, 16, 32, and 64 channels for each of the groups, respectively.

2.3 Activation functions

We will compare the convergence using the Rectified Linear Unit (`relu`) $\hat{x} \mapsto \hat{x}^+$, the hyperbolic tangent (`tanh`) $\hat{x} \mapsto (e^{\hat{x}} - e^{-\hat{x}})/(e^{\hat{x}} + e^{-\hat{x}})$ and sinusoidal (`sine`) $\hat{x} \mapsto \sin(x)$. The latter is a less common choice but has shown superior convergence for wave propagation problems [4]. Weight initialization is a significant step before training the model and depends on the activation function used. Glorot initialization [13] is used for `relu` and `tanh`, whereas the initialization advised in [14] is used for the `sine` activations. Data normalization is done in the spatial dimensions ($[-1, 1]$ for `sine` and `tanh` and $[0, 1]$ for `relu`), where the temporal dimension is normalized with the spatial normalization factor to ensure equal resolutions in all dimensions.

2.4 Loss functions

The mean-squared error (MSE) will be used in all experiments and is the default loss for function regression under the inference framework of maximum likelihood when the target variable is assumed Gaussian

$$\text{MSE} = ||\hat{\mathbf{y}} - \mathbf{y}||^2. \tag{5}$$

The MSE will punish large values more and is a good choice when outliers are less pronounced, which is satisfied for our synthetic training data.

2.5 Data resolution

Deep learning methods typically require large data sets for proper training, which also applies when training the DeepONet. Since the training data for our simulations are created synthetically from high-fidelity spectral-element method (SEM) simulations [15], we can, in principle, create any training data sample set utilizing a simulator. However, it is crucial to find a lower bound since generating data quickly gets intractable when enlarging the domain and/or simulating a broader frequency range – especially when going higher dimensions due to the curse of dimensionality. More extensive training data sets also impact the training time and hardware requirement. Also, training a deep neural network on an unnecessary fine-grained data set is not always advantageous, as we will see.

We can estimate a theoretical lower bound of the sampling rate of the discretized wave-propagation data from the Nyquist Theorem, stating that a given band-limited continuous-time signal can be perfectly reconstructed from its discrete-time signal by ensuring that the

continuous signal is sampled using at least two points per wavelength. The number of spatial points for one dimension is calculated as

$$N_{\text{samples}} = \left\lceil \frac{L}{c/(f_{\max} \times \text{ppw})} \right\rceil, \tag{6}$$

where `ppw` is the number of points per wavelength, and L is the dimension length. Though the sufficient Nyquist sampling rate can be used in theory, the sampling rate for the DeepONet to generalize well and the optimizer to find meaningful optima might require oversampling. The data and the corresponding resolutions for the DeepONet can be divided into several parts with eventually different resolution requirements.

Firstly, we consider the resolution of the sample functions representing the initial conditions used as input to the branch net. Remember that the sensors (the sampling points for each sample function) must be located at equal locations across all the sample functions. Therefore, we cannot exploit any sensor location distribution favoring important samples (e.g., non-zero Gaussian sensors) because the source should be allowed to move freely inside the domain, hence a uniform sensor distribution is the best option. We will investigate the sensitivity from the total number of sensors m in (1) calculated using (6).

Secondly, the spatial/temporal coordinate inputs to the trunk net are considered. Contrary to the sample functions, there are no restrictions on choosing the coordinate distributions for different samples, which enables us to learn the continuous operators of the solution. Various sampling techniques have been explored chiefly for physics-informed neural networks [16]. However, when a training data set with a given distribution has to be manipulated, weighting and re-distributing important points to where the physics is most important (e.g., when the gradient is relatively large or when the field is non-zero) becomes more involved and could require substantial computational efforts for realistic time-dependent problems in complex domains. In this work, we will keep the data distribution determined by the Gauss-Lobotto nodes from the fourth-order Lagrange polynomials on a non-uniform grid used in our SEM solver [15] and instead randomly sample data points corresponding to different grid resolutions for building up the training and validation data sets.

Thirdly, we will investigate the impact of the relative resolution between the temporal and spatial dimensions. From numerical theory, the relation between the temporal resolution Δt and the spatial resolution Δx is given by $\Delta t = \lambda \frac{\Delta x}{c}$ ensuring that the distance the wave has traveled after one time step $\Delta t \times c$ is no longer than the spatial resolution Δx required to resolve the wave propagation. In numerical theory, $0 < \lambda \leq 1$ is the so-called Courant-Friedrichs-Lewy stability condition (CFL) that dictates numerical stability for explicit time-stepping schemes and should be set such that the chosen method is numerically stable. There is no such restriction in deep neural networks as training these relies on global optimization across the parameter domain, and the CFL can be set to the maximum value. In fact, the temporal and spatial resolutions can be set independently as long as the frequencies are well resolved according to the Nyquist Theorem. Hence, the temporal and spatial resolutions can be chosen freely as

$$\Delta x = \frac{c}{f_{\max} \times \text{ppw}_x} \tag{7a}$$

$$\Delta t = \frac{1}{f_{\max} \times \text{ppw}_t}, \tag{7b}$$

with $\text{ppw} \geq 2$. For the gradient descent to find meaningful optima, the speed of sound $c = 1$ m/s is used to ensure equal resolutions for all dimensions.

Lastly, the initial source sample density can be determined using (6). The Nyquist Theorem would tell us the minimum distance between source positions to reconstruct the signal as the source moves freely. However, a finer resolution might be needed for the network not to overfit, as we will investigate in the 'Experiments' section.

2.6 Fourier feature expansions

It is well-known that deep neural networks first learn the lower frequency modes of the data and suffer from learning the higher frequency modes. This phenomenon is known as spectral bias [17, 18]. This problem can, to some extent, be overcome by passing the temporal and spatial coordinates through a Fourier feature mapping that enables a deep FNN to learn the high-frequency modes of the data. In this work, we will experiment with the 'Positional encoding mapping' and the 'Gaussian mapping' from [19]. A third 'ES mapping' has been constructed from an analytical solution. In the following, m is the number of feature expansion terms, and d is the dimension of the input data:

Positional encoding mapping:

$$\gamma(\mathbf{x}) = [\ldots, \cos(2\pi f_j \mathbf{x}), \sin(2\pi f_j \mathbf{x}), \ldots]^T, \quad \text{for } j = 0, \ldots, m-1. \tag{8}$$

The frequencies f_j can be log-linear spaced along each dimension or arbitrarily relative to the fundamental frequency $f_0 \in \mathbb{R}^+$.

Gaussian mapping:

$$\gamma(\mathbf{x}) = [\cos(2\pi \mathbf{B}\mathbf{x}), \sin(2\pi \mathbf{B}\mathbf{x})]^T, \tag{9}$$

where each entry in $\mathbf{B} \in \mathbb{R}^{m \times d}$ is sampled from $\mathcal{N}(0, f_0^2)$ and $f_0 \in \mathbb{R}^+$ is a fundamental frequency empirically chosen.

ES mapping:

$$\gamma(x, y, t) = \left[\ldots, \cos(\Omega t) \cos\left(\frac{\pi n_j}{L_x} x\right) \cos\left(\frac{\pi n_j}{L_y} y\right) \times \cos\left(\frac{\pi n_j}{L_y} y\right), \ldots\right]^T, \text{ for } j = 0, \ldots, m-1, \tag{10}$$

where $n_i \in \mathbb{Z}$ are the wave modes empirically chosen and $\Omega^2 = c^2\pi^2[(d_x/L_x)^2 + (d_y/L_y)^2 + (d_z/L_z)^2]$. The equation is an exact solution to the 3-D wave equation in a rectangular domain of size $L_x \times L_y \times L_z$ with perfectly reflecting boundaries. Sine terms can also be included in the expansion.

3. EXPERIMENTS

We will perform a sensitivity analysis on the convergence and accuracy by varying the DeepONet setup. Evaluating all permutations of all choices will be too exhaustive. Instead, we will determine a base model setup and investigate the sensitivity of various choices covering the impact of 1) activation functions, 2) Fourier feature expansions, 3) mod-MLP networks, 4) batch size, 5) the number of layers and neurons, 6) data resolution, and 7) using a CNN compared to mod-MLP for the branch net.

The base model uses the default MLP with 4 layers of width 1024 for both the branch and trunk net. The ADAM optimizer with learning rate 1e-3 and exponential decay is used together with the MSE loss. The batch size is 64 for the branch net and 100 for the trunk net with 2 `ppw` for the branch net input functions (initial condition), 2 `ppw` for the trunk net temporal dimension, 6 `ppw` for the trunk net spatial dimensions, and the source position density is sampled using 6 `ppw`.

To evaluate the learned model's generalization properties, we ensure that the grid points are (mostly) non-overlapping in the training and validation data sets. The training and validation data have been generated using our SEM solver. The training data is generated using 6 `ppw` for the spatial resolution and 6 `ppw` for the source density corresponding to 1363 source positions. The validation data is generated using a spatial resolution of 5 `ppw` with 5 and 33 source positions.

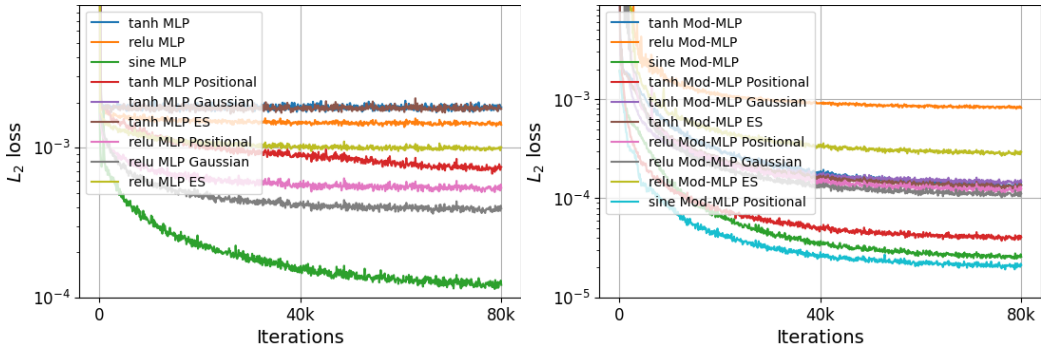
3.1 Activation function

As a first investigation, we will compare the performance using the `tanh`, the `relu`, and the `sine` activation functions for the default MLP architecture with and without Fourier feature expansion. Applying the Fourier feature expansion when the `sine` activation functions are used would be redundant and also show degraded performance (not shown). The L_2 training loss is depicted in Figure 1a (the validation loss is considered in later experiments). We notice that `relu` and `tanh` activation functions are not performing well without Fourier feature expansions. Applying the Fourier feature expansions improves the learning significantly, with the best results obtained for the `relu` activation function with Gaussian expansions. However, using the `sine` activation function outperforms the other choices by a large margin.

In Figure 1b, the same experiments are performed but for the mod-MLP network. We see dramatic improvements for all activations, especially for the `sine` activation choice showing almost an order of magnitude lower L_2 errors. We notice a slight improvement for the `sine` activation when combining the mod-MLP with the positional encodings. It is also interesting to note that the `tanh` activation performs well when positional encodings are applied using the mod-MLP architecture. In the following experiments, we will use the `sine` activation function with positional encodings.

3.2 Batch size

Here, we will investigate the impact of the batch size on the network optimizer to find good optima. The validation data was generated for 5 source positions. In Table 1, the L_2 errors are shown when varying the batch sizes of the branch net function samples from 16 to 96 and the spatial/temporal coordinates for the outputs from 100 to 800. We note that the batch size for the branch net should have size 64 or 96, with corresponding batch sizes of 200 or more for the trunk net. The batch size impacts the computational effort, and a good compromise could be to use a batch size of 64/200 for the branch and trunk net, re-



(a) Default MLP architecture.

(b) Modified MLP architecture.

Figure 1: L_2 validation loss for combinations of activation functions and Fourier feature expansion types. The mod-MLP with `sine` activations dramatically outperforms the other combinations.

Table 1: Batch size experiments using `sine` activation functions. The L_2 training/validation errors are given for each batch size combination, where the total batch size is a multiple of the branch net (BN) sample size and the temporal/spatial batch size for the trunk net (TN). The region with the lowest L_2 errors is highlighted.

Batch sizes branch/trunk net

TN \ BN		Batch size 16	32	64	96
		1e-6×	1e-6×	1e-6×	1e-6×
Batch size	100	7.1/7.2	3.9/5.2	3.0/3.6	2.5/3.4
	200	3.9/5.1	2.7/3.3	**2.2/2.9**	**2.4/2.6**
	400	3.1/4.3	2.0/3.1	**2.0/2.9**	**2.1/2.9**
	800	2.3/3.7	1.9/3.1	**1.9/2.8**	**2.0/2.9**

spectively, which we have chosen in the following experiments.

3.3 Number layers and neurons

Table 2 shows the L_2 errors when varying the network depth from 3 to 5 and the network width from 512 to 2048. We observe that wider networks are more critical for achieving good results than deeper ones. Using `sine` activation function can be seen as a Fourier series expansion [20], which could be the reason for the better performance using wider networks. Applying wider networks

Table 2: L_2 errors for varying the number of layers (L) and neurons (N) using the sine activation function.

Layers/neurons architectures

L \ N		512	1024	2048
2	t	1.9e-5	2.8e-6	**1.6e-6**
	v	1.9e-5	4.4e-6	**3.2e-6**
3	t	6.8e-6	**2.0e-6**	**1.5e-6**
	v	7.6e-6	**3.0e-6**	**3.0e-6**
4	t	4.6e-6	**2.0e-6**	**1.8e-6**
	v	5.4e-6	**2.9e-6**	**2.4e-6**
5	t	4.2e-6	**2.0e-6**	**1.6e-6**
	v	4.8e-6	**2.9e-6**	**2.4e-6**

is more computationally expensive; therefore, choosing a compromise with more layers with fewer neurons can be necessary. We will use the wider layers with 2048 neurons and 2 layers, although slightly better results are obtained with 4 layers.

3.4 Data resolution

The validation data for the following experiments includes 33 source positions instead of the five source position for the previous two experiments to get a more truthful picture of generalization when varying the data resolutions. In Subtable 3a, results are shown for combinations of function resolutions for the branch net of 2 and 4 `ppw` with spatial data resolutions for the trunk net of 2, 4, and 6 `ppw`

Table 3: Data resolution for a 2/2048 layers/neurons MLP for the branch and trunk net and batch size 64/200.

Relative resolution branch/trunk net

BN \ TN		PPW 2	PPW 4	PPW 6
PPW	2	1.5e-6/1.1e-4	**1.9/6.1e-6**	**1.9/5.5e-6**
	4	1.6e-6/6.3e-5	1.9/6.0e-6	1.7/5.1e-6

(a) Data resolution combinations for the branch net (BN) and trunk net (TN).

Relative resolution src density/space/time

PPW Δx_{srcpos}	Δx	Δt	train	val
2	6	2	1.5e-6	1.7e-5
3	6	2	1.7e-6	7.5e-6
4	6	2	1.7e-6	6.1e-6
5	6	2	**1.7e-6**	**5.0e-6**
6	6	2	**1.8e-6**	**5.3e-6**
6	6	4	**2.0e-6**	**4.8e-6**

(b) Impact from the source density on the generalization properties.

while keeping the temporal resolution fixed at 2 ppw. We notice severe overfitting when only 2 ppw are used for the spatial resolution, with 6 ppw giving the best results. No significant impact is observed when varying the input function resolution of the branch net. In Subtable 3b, the influence from the sampling density of the source positions Δx_{srcpos} is shown, and we notice the least overfitting when 5 or 6 ppw are used. We also note that using 4 ppw for the temporal resolution is not increasing the accuracy by much. This is true since we are overfitting to the uniform temporal steps when only using 2 ppw, which is not a problem since temporal interpolation is not of practical interest.

3.5 Convolutional neural network for the branch net

Finally, we will compare two ResNet architectures, the ResNet-{3,3,3,3} with {16,32,64,128} hidden channels for input functions of size 18×18 sampled with 2 ppw and the ResNet-{3,3,3,3,3} with hidden channels {16,32,64,128,256} for input functions of size 35×35 sampled with 4 ppw. The standard ReLU activation function is used for the CNN architecture. A single linear output MLP layer is used with sine activation functions. The convergence of the loss can be seen in Figure 2 plotted together with the mod-MLP for comparison and shows similar performance for all ResNet architectures on par with the mod-MLP architecture reference.

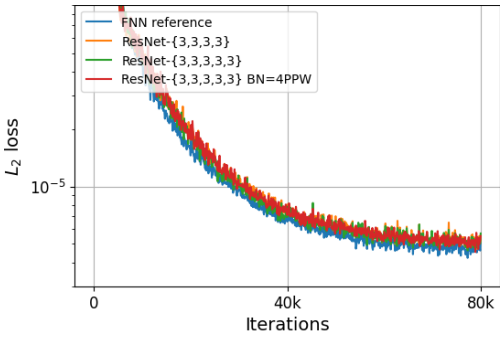


Figure 2: Validation loss for ResNet-{3,3,3} and ResNet-{3,3,3,3} for branch input function resolutions of 2 and 4 ppw.

4. CONCLUSION

We have systematically studied the DeepONet sensitivity for wave propagation problems, focusing on network architectures, data fidelity, and operator learning parameters. The most prominent choice for successfully training the model is to use the mod-MLP architecture in combination with sine activation functions. Moreover, the spatial data resolution greatly impacts the accuracy of the trained model, where 4-6 ppw for the spatial and 5-6 ppw for the source density resolutions are required for the network to generalize well for unseen source and receiver positions.

5. ACKNOWLEDGMENTS

Thanks to DTU Computing Center GPULAB for access to GPU clusters and especially to Sebastian Borchert for his assistance. Also, thanks to Finnur Pind for access to an SEM code for data generation. Thanks to George Em Karniadakis for an inspiring stay with the CRUNCH group.

6. REFERENCES

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equa-

tions," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[2] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[3] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: a review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021. http://arxiv.org/pdf/2105.09506.

[4] N. Borrel-Jensen, A. Engsig-Karup, and C.-H. Jeong, "Physics-informed neural networks for one-dimensional sound field predictions with parameterized sources and impedance boundaries," *Jasa Express Letters*, vol. 1, no. 12, 2021.

[5] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[6] "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, pp. 218–229, 2021.

[7] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.

[8] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *CoRR*, vol. abs/2010.08895, 2020.

[9] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[10] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed deeponets," 2021.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9, pp. 249–256, PMLR, 13–15 May 2010.

[14] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," 2020.

[15] F. Pind, A. P. Engsig-Karup, C.-H. Jeong, J. S. Hesthaven, M. S. Mejling, and J. Strømann-Andersen, "Time domain room acoustic simulations using the spectral element method," *The Journal of the Acoustical Society of America*, vol. 145, no. 6, pp. 3299–3310, 2019.

[16] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115671, Jan. 2023.

[17] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," 6 2018.

[18] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman, "Frequency bias in neural networks for input of non-uniform density," 3 2020.

[19] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," 6 2020.

[20] N. Benbarka, T. Hofer, H. Ul-Moqeet Riaz, and A. Zell, "Seeing Implicit Neural Representations as Fourier Series," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, (Waikoloa, HI, USA), pp. 2283–2292, IEEE, Jan. 2022.

Paper D

Accelerated sound propagation simulations using an error-free Fourier method coupled with the spectral element method

Nikolas Borrel-Jensen[1]
Acoustic Technology, Department of Electrical and Photonics Engineering, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

Allan P. Engsig-Karup[2]
Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

Maarten Hornikx[3]
Department of the Built Environment, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands

Cheol-Ho Jeong[4]
Acoustic Technology, Department of Electrical and Photonics Engineering, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

ABSTRACT

Simulating acoustics efficiently and accurately using numerical methods has been an active research area for the last decades and has applications in computer games, VR/AR, and architectural design. However, their extensive computation time makes these methods challenging for large scenes and broad frequency ranges. This work attempts to accelerate the simulations using rectangular decomposition, enabling error-free propagation in the bulk of the domain consisting of air. We exploit the analytical solution to the wave equation in rectangular domains calculated using the Fast Fourier Transform with near-optimal spatial discretization satisfying the Nyquist criterium. Coupling with the spectral element method near the boundaries results in a method capable of handling complex geometries with realistic boundaries, though with the caveat that additional errors and computational overhead may result from the interface. This paper will investigate the accuracy and efficiency of the proposed domain decomposition method compared to a spectral element implementation running in the entire domain and the results in 1D indicate an 18 times speedup factor for relative errors below 9%.

[1]nibor@dtu.dk

[2]apek@dtu.dk

[3]m.c.j.hornikx@tue.nl

[4]chje@dtu.dk

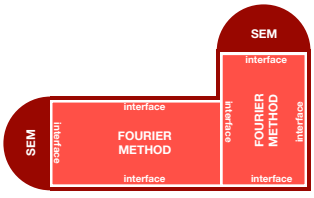
1. INTRODUCTION

Simulating acoustics in virtual spaces is an active research topic, with applications in computer games, VR/AR, and building acoustics. Using numerical methods to solve the underlying partial differential equations naturally takes the wave phenomena into account. However, it comes with the cost of being computational demanding, especially when simulating broad frequency ranges and large domains. To overcome the demanding computations, we will develop an efficient and accurate domain decomposition method coupling an efficient and error-free Fourier method in rectangular domains with an spectral element method applied at the boundaries handling complex geometries and impedance boundaries.

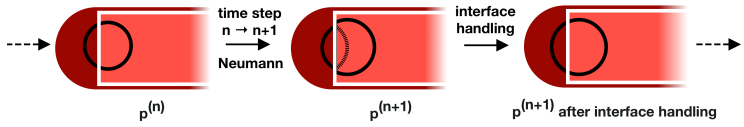
The most used numerical methods for simulating acoustics are the finite element methods (FEM) [12], spectral element methods (SEM) [14, 19], DG-FEM [11], finite-difference time-domain methods (FDTD) [5], boundary element methods [6], the Wave-Based Method (WBM) [17], and Pseudo-spectral Fourier methods [8]. Domain decomposition methods (DDM) [3] are widely used approaches where the domain is split into many partitions. We will use DDMs to divide the domain into simpler partitions where specialized methods can be applied independently depending on the properties of the domain. Our approach is inspired by Raghuvanshi et al. [10, 16], where an adaptive rectangular decomposition (ARD) method was proposed, exploiting the well-known analytical solution to the wave-equation in rectangular domains coupled with the FDTD method near the boundaries. The ARD method has error-free propagation within the rectangular domains and can be computed very efficiently using the Fast Fourier Transform (FFT). Only numerical errors are introduced at the interfaces. We follow the same approach, but instead of coupling with FDTD, we couple the Fourier method (FM) with SEM, still using FDTD for interface handling though. The SEM is advantageous over FDTD as it handles complex geometries and impedance boundaries [14] in a robust manner. By coupling the efficient Fourier method running in rectangular domains, and the SEM near the boundaries, we can speed up the overall computation time for large scenes, where the ratio between air and boundaries is large. Related domain decomposition approaches have been taken to achieve similar goals. In [13], the time-domain Pseudo-Spectral element method was coupled with the DG-FEM method near the boundaries. In [18] the Wave-Based method is applied in convex domains and coupled with the second-order SEM in domains requiring geometrical flexibility.

We propose a method consisting of three parts: 1) an error-free Fourier method (FM) running in rectangular domains, 2) the SEM for modeling complex geometries and realistic impedance boundaries, and 3) an FDTD scheme for interface handling. In Figure 1a the domain decomposition is illustrated.

Figure 1: Overview of the domain decomposition methods coupling the Fourier method in rectangular partitions with the spectral element method near the boundaries.



(a) A domain decomposed into rectangular partitions running the error-free FM coupled with the SEM near the boundaries.



(b) Time step from n to $n+1$ with Neumann boundary conditions at the interface for each of the methods running in the separate partitions. Note, that the methods in the partitions are running independently and compensation of the reflections at the interface is done after each time step.

2. GOVERNING EQUATIONS

The wave equation is

$$\frac{\partial^2 p(\mathbf{x},t)}{\partial t^2} - c^2 \nabla p(\mathbf{x},t) = F(\mathbf{x},t), \qquad t \in \mathbb{R}^+, \qquad \mathbf{x} \in \mathbb{R}^N. \tag{1}$$

where p is the pressure (Pa), t is the time (s) and c is the speed of sound in air (m/s) and $F(\mathbf{x},t)$ is a forcing function. An initial conditions can be used instead of a forcing function satisfied by using e.g. a Gaussian source for the pressure part and setting the velocity equal to zero

$$p(\mathbf{x}, t=0) = \exp\left[-\left(\frac{\mathbf{x}-\mathbf{x}_0}{\sigma_0}\right)^2\right], \qquad \frac{\partial p(\mathbf{x}, t=0)}{\partial t} = 0, \qquad \mathbf{x}_0 \in \mathbb{R}^N, \tag{2}$$

with σ_0 being the width of the pulse determining the frequencies to span and boundary conditions

$$\frac{\partial p(\mathbf{x},t)}{\partial \mathbf{n}} = \overline{v}, \qquad \mathbf{x} \in \Gamma_v, \tag{3}$$

where $\overline{v}$ is the enforced velocity at the boundary Γ_v, and $\mathbf{n}$ is the normal pointing outwards from the boundary Γ. Equation (3) is the Neumann boundary conditions and is applied at the interface for coupling domains – as explained in Section 4 – with $\overline{v} = 0$.

3. THE FOURIER METHOD

3.1. The analytical solution

It can be shown by using separation of variables [1] (p. 155-157), that any 1D pressure field $p(x,t)$ in *rectangular domains* with *Neumann boundary conditions* can be represented in the form of a general series representation

$$p(x,t) = \sum_{i=0}^{N-1} M_i(t)\Phi_i(x), \qquad \text{where} \quad M_i(t) = ae^{jck_it} + be^{-jck_it}, \quad k_i = \pi\frac{i}{l}, \tag{4}$$

and i is the mode, N is the maximum mode to include depending on the required frequency range, $M_i(t)$ is the time-varying mode coefficients [9, 15], $j = \sqrt{-1}$ is the complex number, k_i is the wavenumber, and $\Phi_i(x) = \cos(k_i x)$ are the eigenfunctions of the Laplacian for a rectangular domain. The time constants a and b depend on the initial conditions.

3.2. The discrete formulation

We can interpret the analytical solution (4) on a discrete uniform grid x_i with the highest wavenumber being spatially sampled at the Nyquist rate. Assuming that the signal is properly band-limited and that enough modes N are included to capture the band-limited signal, the discretization introduces no numerical errors.

In the discrete interpretation, Equation (4) is similar to the inverse Discrete Cosine Transform, with ϕ_i being the cosine basis vectors. Hence, we can convert from modal coefficients $\mathbf{M}$ to pressure values $\mathbf{p}$ as

$$\mathbf{p}(t) = \text{iDCT}(\mathbf{M}(t)), \tag{5}$$

where $\mathbf{p}$ is the $N \times 1$ pressure vector and $\mathbf{M}(t)$ is the $N \times 1$ vector including all modes for time t. Reinterpreting the wave equation $\frac{\partial^2 p}{\partial t^2} - c^2\frac{\partial^2 p}{\partial x^2} = F$ in a discrete setting for the spatial dimensions and taking the cosine transform of both sides of the equality sign yields

$$\frac{\partial^2}{\partial t^2}\mathbf{M}(t) + c^2\mathbf{k}^2 \odot \mathbf{M}(t) = \text{DCT}(\mathbf{F}(t)), \tag{6}$$

where $\mathbf{F}(t)$ is a $N \times 1$ vector with pressure values of the forcing function in all spatial nodes at time t, $\mathbf{M}(t)$ is a $N \times 1$ vector with mode coefficients corresponding to time t, $\mathbf{k}$ is a $N \times 1$ vector including the wave numbers, and $\odot$ is the Hadamard product operator. Disregarding the forcing term, the above equation describes a set of independent simple harmonic oscillators, each vibrating with its characteristic frequency $\omega_i = ck_i$.

In the case where the forcing term is a constant transformed into mode-space $\tilde{F}$, we can simply solve the equation $\frac{\partial^2}{\partial t^2} M_i + \omega_i^2 M_i - \tilde{F}_i = 0$, but since the forcing term changes with time, we need to derive a temporal update scheme for this. We will assume that the forcing term is constant over a time-step Δt, which is satisfied when proper Nyquist sampling is applied. The forcing term may be transformed into mode-space as $\tilde{\mathbf{F}}(n\Delta t) \triangleq \mathrm{DCT}(\mathbf{F}(n\Delta t))$. We will use the second-order centered difference multiplied by the term $\frac{\omega^2 \Delta t^2}{2(1-\cos(\omega \Delta t))}$ originating from the solution to the simple harmonic oscillator, obtaining the update scheme [16]

$$M_i^{(n+1)} = 2M_i^{(n)} \cos(\omega_i \Delta t) - M_i^{(n-1)} + \frac{2\tilde{F}^{(n)}}{\omega_i^2}(1 - \cos(\omega_i \Delta t)). \tag{7}$$

3.3. Discrete Fourier Transform for imposing Neumann boundary conditions

The formulation in the analytical formulation in (4) is real and hence, to ensure a real-valued frequency spectrum after applying the Discrete Fourier Transform, an even Fourier continuation $p_k = p_{-k}$ and periodicity[5] are required. Mirroring the samples around a boundary satifies the zero derivative Neumann boundary condition in that point. The extended time signal $\hat{p}_k$ of lenght $(2N-2)$ is then

$$\hat{p}_k = \begin{cases} p_k, & \text{for} \quad 0 \leq k < N, \\ p_{2N-k}, & \text{for} \quad N \leq k < 2N-1. \end{cases} \tag{8}$$

The single-sided frequency spectrum M_i consisting of N bins is used for time-stepping in Equation (7). Again, but now in the Fourier domain, the even and periodic double-sided spectrum $\hat{M}_i$ of size $(2N-2)$ is reconstructed and the inverse Fourier Transform is applied to retrieve the time pressures back as

$$\hat{M}_i = \begin{cases} M_i, & \text{for} \quad 0 \leq i < N, \\ M_{2N-i}, & \text{for} \quad N \leq i \leq 2N-1. \end{cases} \tag{9}$$

Extracting the first N pressures values from $\mathcal{F}^{-1}(\hat{M}_i)$ gives us the final result. Hence, the DCT and iDCT used in Section 3 and the remaining sections are referring to the definitions below

$$\begin{aligned} \mathrm{DCT}\left[p_k\right] &= \mathrm{Re}\left(\mathcal{F}[\hat{p}_k]\right), \\ \mathrm{iDCT}[M_i] &= \mathrm{Re}\left(\mathcal{F}^{-1}[\hat{M}_i]\right). \end{aligned} \tag{10}$$

4. INTERFACE HANDLING

We will derive the interface scheme for communicating pressure values between partitions following the same approach as Raghuvanshi [15, 16] to couple the FM in rectangular partitions with the SEM near the boundaries. For clarity of presentation, we will consider the 1D wave equation using the second-order centered differences. The wave equation (1) discretized in space can be written as $\frac{\partial^2 P}{\partial t^2} - KP = F(t)$, where K represents the Laplacian operator at each node as a Discrete Laplacian Matrix and F is the forcing term at each node. As a result of spatial discretization, K is transformed

[5] if we have a signal 'abcd', then the even extension would be 'abcdcb'.

into a symmetric matrix

$$K = \frac{c^2}{\Delta x^2}\begin{bmatrix} \ddots & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & & \ddots \end{bmatrix}. \tag{11}$$

Assume that the grid consists of N nodes. Care must be taken at the boundary, since calculating the pressure value p_N at the right-most node N would require the neighboring pressure value p_{N+1} located outside the grid for a Neumann boundary condition imposed by mirroring the pressure halfway between the nodes or at the boundary node. We will denote the pressure values at these ghost nodes as the *residual*. Eliminating the ghost nodes happens by discretizing the BCs and inserting these expressions into the scheme. For example, for a right boundary node x_N, the residual is added to the inner partition as

$$\begin{aligned} p''_{N+1/2}(x_N) &= \frac{p(x_{N-1}) - 2p(x_N) + p(x_{N+1})}{\Delta x^2} \overset{\text{N+1/2}}{\overset{\text{Neumann}}{=}} \frac{p(x_{N-1}) - p(x_N)}{\Delta x^2}, \\ p''_N(x_N) &= \frac{p(x_{N-1}) - 2p(x_N) + p(x_{N+1})}{\Delta x^2} \overset{\text{N}}{\overset{\text{Neumann}}{=}} \frac{2p(x_{N-1}) - 2p(x_N)}{\Delta x^2}. \end{aligned} \tag{12}$$

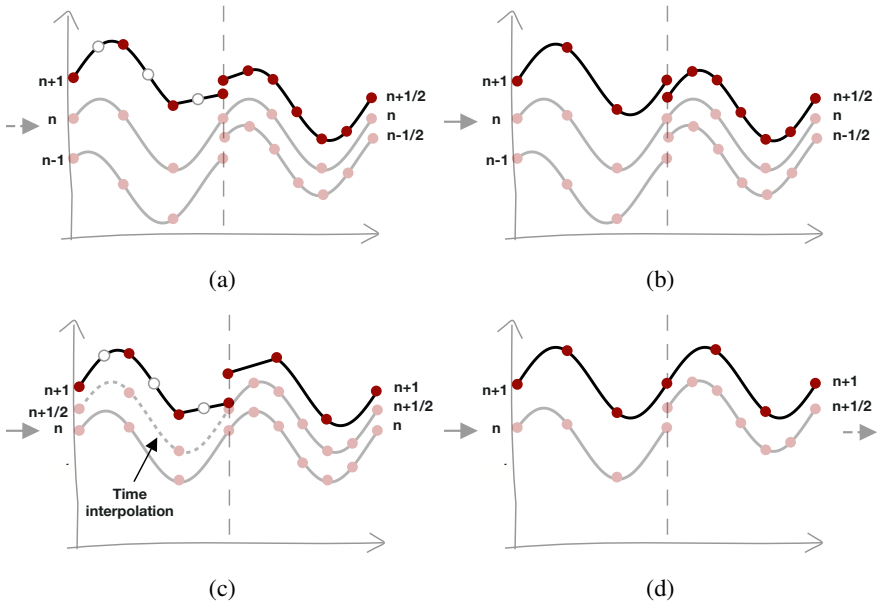
These observations can be used to formulate the interface handling between two separate partitions. Suppose we wish to split the domain into two equal partitions with N nodes each, such that each might be treated independently. P is a vector of length $2N$. This can be done by decoupling K into a block diagonal form while accounting properly for the off-diagonal entries $K = A + C$, where the decoupled matrix A and the coupling matrix C are given by (interface located at $N + 1/2$)

$$A = \frac{c^2}{\Delta x^2}\left[\begin{array}{ccc|ccc} \ddots & & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -1 & & & \\ \hline & & & -1 & 1 & \\ & & & 1 & -2 & 1 \\ & & & & & \ddots \end{array}\right], \qquad C = \frac{c^2}{\Delta x^2}\left[\begin{array}{ccc|ccc} 0 & & & & & \\ & \ddots & & & & \\ & & -1 & 1 & & \\ \hline & & 1 & -1 & & \\ & & & & \ddots & \\ & & & & & 0 \end{array}\right]. \tag{13}$$

The pressure is first updated using A, and the residual part not taken into account is then computed using C. The coupling matrix C can intuitively be seen as a communication step to pass pressures between two partitions, and the action of C is to compute the gradient of the pressure at the interface (scaled by some factor). Rewriting in terms of the decoupled and coupled matrices yields $\frac{\partial^2 P}{\partial t^2} - AP = F(t) + CP$, where the coupling term CP can be seen as an additional source term accounting for the alignment of pressures between partitions as $\hat{F}(t) = F(t) + CP$.

We can formulate a unified framework for communicating pressures between partitions running *any* method. The boundary condition will be imposed at the pressure node N (not $N + 1/2$ as above) since this will ease the implementation for the SEM where boundaries are explicitly defined at the nodal points. Without lack of generality, assume that two independent (2,2) FDTD update schemes are running in each partition and denote the pressures in partition 1 as $p_{1,i}$ and partition 2 as $p_{2,i}$ with subscripts denoting the partition and corresponding node index, respectively. Then, the interface communication can be handled as follows:

Figure 2: Interface handling with twice refined spatial and temporal resolutions in the right partition compared to the left partition. (a) Neumann reflections present at time $n+1$ and $n+1/2$ with spatial interpolation in partition 1. (b) Interface handling for time $n+1$ and $n+1/2$ by using the corresponding Δx_1, Δt_1, Δx_2, Δt_2 values for the FDTD scheme in Equation (14)-(16). (c) Temporal interpolation in the coarse domain for calculating $n+1$. (d) Interface handling for the fine domain (only) at time $n+1$.



1. Calculate the pressures in each domain as completely independent partitions with Neumann boundary condition at the interface

$$
\begin{aligned}
p_{1,N}^{(n+1)} &\leftarrow \frac{c^2\Delta t_1^2}{\Delta x_1^2}\left(2p_{1,N-1}^{(n)} - 2p_{1,N}^{(n)}\right) + 2p_{1,N}^{(n)} - p_{1,N}^{(n-1)} + F_{1,N}^{(n)}, \\
p_{2,1}^{(n+1)} &\leftarrow \frac{c^2\Delta t_2^2}{\Delta x_2^2}\left(-2p_{2,1}^{(n)} + 2p_{2,2}^{(n)}\right) + 2p_{2,1}^{(n)} - p_{2,1}^{(n-1)} + F_{2,1}^{(n)}.
\end{aligned}
\tag{14}
$$

2. Remove the residual part at time n corresponding to the reflected pressures

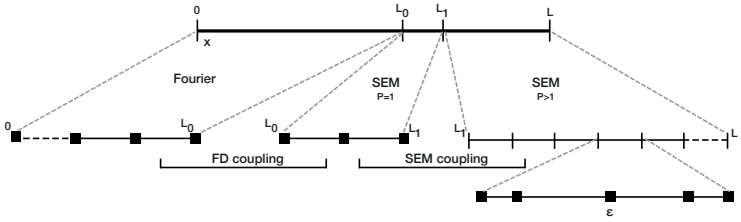
$$
p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} - \frac{c^2\Delta t_1^2}{\Delta x_1^2}\left(p_{1,N-1}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} - \frac{c^2\Delta t_2^2}{\Delta x_2^2}\left(p_{2,2}^{(n)}\right). \tag{15}
$$

3. Transfer the removed residual part at time n to the neighboring partition(s)

$$
p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} + \frac{c^2\Delta t_1^2}{\Delta x_1^2}\cdot\left(p_{2,2}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} + \frac{c^2\Delta t_2^2}{\Delta x_2^2}\left(p_{1,N-1}^{(n)}\right). \tag{16}
$$

Separating the pressure update scheme inside the cavity from the interface handling scheme makes it clear that we can transfer pressures between partitions running any schemes. The sixth-order update scheme used in this work can be derived in a similar manner. The spatial and temporal resolutions will differ between partitions if the efficiency of the Fourier method is to be exploited and therefore piece-wise cubic Hermite interpolation is used in both space and time [2]. The procedure is depicted in Figure 2.

Figure 3: Domain decomposition coupling FM with SEM by introducing a first-order SEM layer to stabilize the scheme.



4.1. Interfacing with the spectral element method

Since the interface handling is independent of the methods running in the partitions, coupling the FM with the SEM can be implemented using the framework described. However, due to the non-physical behavior caused by the Neumann condition at the interface at each time-step before compensating, non-smooth second derivatives are introduced. This type of behavior is known as 'shocks' in the literature, and it is well-known to introduce challenges for the SEM. An investigation has been made by comparing the Laplacian term from Equation (1) for the SEM and FDTD methods

$$\text{(SEM)} \quad \mathbf{p}^{(n+1)} = 2\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)} \overbrace{\boxed{-c^2\Delta t^2 \mathcal{M}^{-1}\left(\mathcal{S}\mathbf{p}^{(n)}\right)}}^{\text{SEM Laplacian}} + \Delta t^2 \mathbf{f}^{(n)}, \tag{17}$$

$$\text{(FDTD)} \quad \mathbf{p}^{(n+1)} = 2\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)} \overbrace{\boxed{+\frac{c^2\Delta t^2}{\Delta x^2} K\mathbf{p}^{(n)}}}^{\text{FDTD Laplacian}} + \Delta t^2 \mathbf{f}^{(n)}, \tag{18}$$

where $\mathcal{M}$ is the mass matrix and $\mathcal{S}$ is the stiffness matrix [4, 7]. We have noted that noticable bigger errors are introduced for higher-order SEM compared to first-order SEM. A simple remedy was to add an SEM layer of first-order polynomials near the interface as illustrated in Figure 3.

5. NUMERICAL EXPERIMENTS

We will investigate the accuracy and efficiency of the domain decomposition method coupling the FM with the SEM. All experiments are done in a 1D domain with $f_{\max} = 1000$ Hz with the differentiated Gaussian pulse injected halfway into the FM partition 1. In the following, we will use the notation FM{ppw} and SEM{ppw}, i.e., FM4-SEM8 would be the FM-SEM coupling with spatial resolutions of four and eight points per wavelength (ppw) per partition, respectively.

To assess the accuracy of the method, we will separately consider interface errors and errors over time at a receiver position. The interface errors are investigated by measuring the reflected pressures in the vicinity left to the interface for a left to right traveling wave at time $t_{\text{refl}} = 0.0135$ s corresponding to the wave having made a single pass through the interface. The errors over time at the receiver position is compared against a FM reference solution for a total running time $t_{\max} = 0.2$ s using different measures.

5.1. Interface errors

The domain length in this experiment is 10 m split into two partitions of 5 m each. We measure the interface pressure errors in dB as

$$\epsilon_{\text{interface}} = 20\log_{10}\left(\frac{\max(|\mathbf{p}_{\text{interface}}|)}{|p_{\text{inc}}|}\right) \text{ dB}, \tag{19}$$

where $\mathbf{p}_{\text{interface}} = \{p(t,x) \mid x \in [3.5, 5]m,\ t = 0.0135s\}$ are the reflected pressure values in the vicinity of the interface after a single interface traversal and has been chosen such that no other wavefronts

Table 1: Errors for the FM-SEM coupled using a sixth-order FDTD scheme for different spatial resolutions determined by the points per wavelength, keeping the frequency range fixed. (a) FM-SEM interface errors,(b) Relative error ϵ_{rel} and maximum error ϵ_{∞} calculated by comparing with an exact Fourier reference solution.

Interface errors at $t = 0.0135$ s

FM \ SEM	3	4	6	8	9	12
3	-25 dB	-26 dB†	-28 dB	-26 dB†	-26 dB	-25 dB
4	-	-34 dB	-35 dB†	-36 dB	-32 dB†	-32 dB

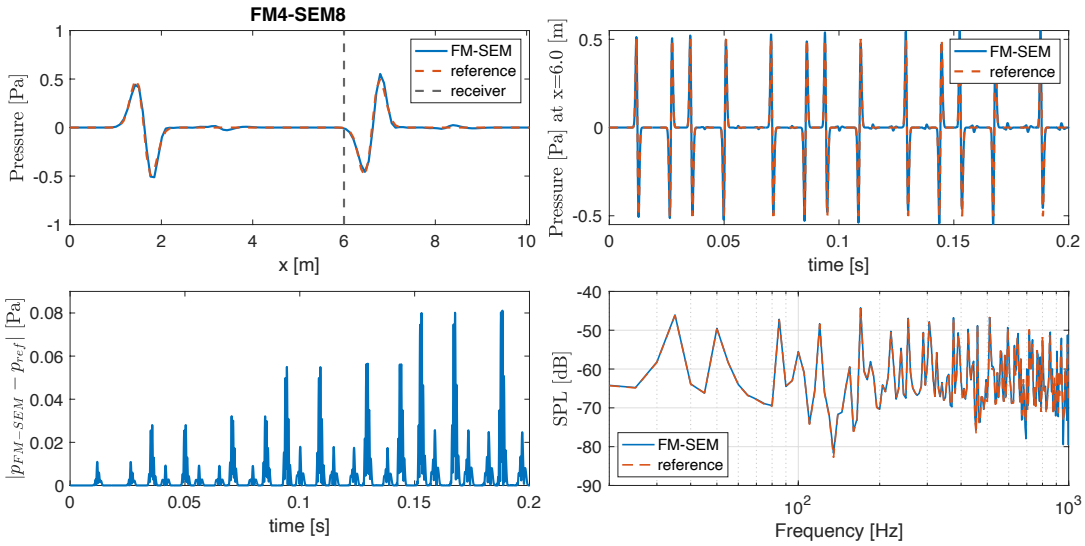
(a) †: $\Delta t_1 = \Delta t_2$, since the spatial resolution is not a multiple of each other.

Errors at $t = 0.2$ s

FM↔SEM [ppw]	ϵ_{rel}	ϵ_{∞}
$4 \leftrightarrow 4$	-	0.2528
$4 \leftrightarrow 8$	9.1%	0.0811
$4 \leftrightarrow 12$	5.7%	0.0562

(b)

Figure 4: FM4-SEM8 running for $t_{\text{max}} = 0.2$ s. From left to right: 1) Wave propagation in the full domain at $t = 0.2$ s, 2) impulse response at receiver position $x = 6.0$ m, 3) L_1 IR error, 4) transfer function at $x = 6.0$ m.



are overlapping. The pressure values before traversing the interface are normalized by the incident pressure p_{inc} in the domain. $\epsilon_{\text{interface}} = -\infty$ corresponds to zero pressures (no interface error) being reflected.

The interface errors for the FM-SEM coupling are summarized in Table 1a. The errors are measured for combinations of spatial resolutions in the partitions denoted by the number of ppw. Overall, we see that using three ppw for the FM shows large errors indicating that four ppw is the lowest resolution possible for reasonable accuracies. The FM can handle two ppw corresponding to the Nyquist criterium but is limited by the sixth-order FD scheme at the interface. FM4-SEM8 gives the lowest interface error of -36 dB.

5.2. Overall accuracy

Another investigation of the accuracy is to compare the simulation with a reference solution. The error of the impulse responses (IRs) at receiver position $x = 6.0$ m is calculated in the relative error ϵ_{rel} and the maximum error ϵ_∞ norms

$$\epsilon_{\text{rel}}(x) = \frac{1}{T}\sum_{n=0}^{T-1}\frac{|p_{\text{sim}}(t_n, x) - p_{\text{ref}}(t_n, x)|}{|p_{\text{ref}}(t_n, x)|}, \qquad \epsilon_\infty(x) = \max_{n=0,1,\ldots,T-1}|p_{\text{sim}}(t_n, x) - p_{\text{ref}}(t_n, x)|. \tag{20}$$

The simulated pressures $p_{\text{sim}}(t_n, x)$ and the reference pressures $p_{\text{ref}}(t_n, x)$ correspond to the IRs at the receiver position x for the discrete time steps t_n, and $T = \lceil t_{\max}/\Delta t \rceil$ is the total number of time steps. The results are summarized in Table 1b. First, we notice that FM4-SEM4 gives the largest errors $\epsilon_\infty = 0.25$ and very big relative errors (not included in the table). In contrary, coupling with SEM8 or SEM12 drastically improves the accuracy with errors of $\epsilon_\infty < 0.08$ and $\epsilon_{\text{rel}} < 9\%$. These results are not all consistent with the interface errors; notice for example, that the FM4-SEM8 coupling (-36 dB) compared to FM4-SEM12 (-32 dB) has 4 dB lower reflection errors, which could indicate that the SEM is introducing numerical dispersion errors also contributing to the overall error. Using other time integration schemes, such as the Runge-Kutta method, should decrease the numerical SEM errors.

In Figure 4, the FM4-SEM8 is plotted against a reference solution showing the domain pressure at $t_{\max} = 0.2$ s, the IR and the transfer function (TF) at receiver position $x = 6.0$ m, and the corresponding L_1 errors over time for the IR. We see a good match between the simulation and the reference, though some small noticeable pressure perturbations are primarily due to interface errors.

5.3. Convergence

In Figure 5 the convergence is plotted pair-wise for FM{2,4,8,16,32}-SEM{4,8,16,32,64} with 1) fixed temporal resolution, 2) individual temporal resolution satisfying the Courant-Friedrichs-Lewy condition, and 3) pair-wise for FM{16,32}-SEM{16-2,32-4} where the SEM $P = 1$ interface layer is oversampled eight times compared to the main SEM with fixed temporal resolution. When the $P = 1$ layer is finely oversampled, the forth-order spectral convergence is preserved, indicating that the interface errors are converging at the same rate.

5.4. Efficiency

The theoretical speedup in 3D when running FM instead of SEM in the full domain is $2 \times r^3$, where r is the spatial resolution factor between the FM and the SEM running in the two partitions, resulting in a 16× speedup for FM4-SEM8 and 54× speedup for FM4-SEM12. The factor of 2 stems from the time resolution being twice as coarse for the FM. On top of that, the SEM time complexity for solving the system of equations consisting of sparse band matrices can be done in $O(q^2 n) + O(qn)$ in time using direct solvers, where q is the bandwidth of the matrix and n is the degrees of freedom. The Fourier method is $O(N \log(N))$ in time when using the Fast Fourier Transform.

We will perform an empirical evaluation of the efficiency gained from the FM-SEM coupling compared to running the SEM in the entire domain for $t_{\max} = 0.2$ s. The methods are implemented

Figure 5: h-Convergence plots: FM{2,4,8,16,32}-SEM{4,8,16,32,64} are plotted pair-wise in the two top-most graphs, where 1) the temporal resolution is fixed for the blue graph, and 2) the temporal SEM resolution is twice the FM resolution for the blue graph. The lower red graph 3) shows convergence according to polynomial order $P = 4$ when oversampling 8 times the SEM $P = 1$ interface layer compared to the main SEM $P = 4$.

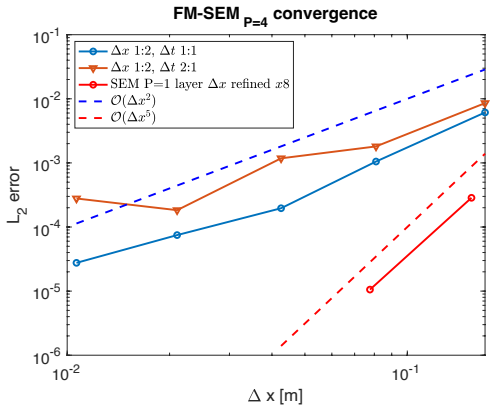


Table 2: FM4-SEM8 CPU timings for $l = 50$ m

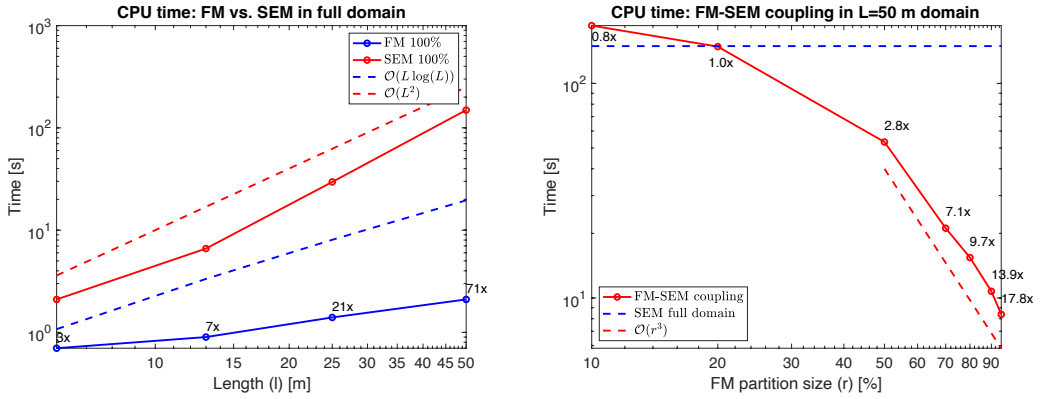
FM partition size (r)	FM solver	SEM solver	interface
50%	1.2%	84.9%	13.0%
80%	6.0%	53.6%	39.6%
95%	10.1%	18.5%	70.3%

in Matlab, and the timings exclude the matrix assembly. The vast majority of the time in the SEM is spent solving the linear system of equations and is implemented using the Matlab backslash operator $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$ for solving the linear system $\mathbf{Ax} = \mathbf{b}$. For the FM, most of the work is spent in the Fourier transformation, where the build-in Matlab function 'FFT' is used. In all experiments, we will compare against the baseline forth-order SEM with Neumann boundaries. Future work should investigate more optimal solvers – such as sparse solvers – for the SEM to make the comparison more fair.

In the first experiment, we compare the CPU time for a 1D case separately for the FM and SEM running in the entire 1D domain of lengths $l = [6, 13, 25, 50]$ m, i.e., with no couplings. The result is shown in Figure 6a, and we see 3x to 71x speedups depending on the domain size. The CPU times scale with $O(l^2)$ for the SEM and below the theoretical $O(l \log(l))$ limit for the FM.

In the second experiment, we fix the 1D domain size to 50 m and compare the CPU time for the FM-SEM coupling for different Fourier partition sizes of $r = [10, 20, 50, 70, 80, 90, 95]$ % relative to the entire domain. In Figure 6b the results are depicted, and we achieve speedups between 2x and 17x for Fourier partition sizes above 50% with scaling close to $O(r^3)$. In Table 2, the timings for $l = 50$ m are shown separately for the FM and SEM solvers and the interface handling. We notice that the SEM workload for a 50/50 partition split is taking 85% of the total computation time. Increasing the partition size of FM drastically decreases the SEM workload, and for $r = 95$ %, the workload of the interface handling starts dominating, taking up 70% of the time. Most of the workload at the interface is because of the space and time spline interpolation, though there is significant overhead when calling the interpolation methods interpolating only a few points near the interface. In fact, interpolating all pressure values instead of only the values around the interface has a minor impact on the absolute performance. Therefore, we expect more time-efficient interpolations when extending to 2D and 3D, where much bigger pressure grids are to be interpolated near the interface.

Figure 6: CPU timings in a 1D domain. a) Comparison between the FM and the SEM running in the full domain for different domain sizes (no couplings), b) Comparison between FM-SEM and a baseline SEM with different relative FM partition sizes.



(a) (b)

CONCLUSION

We have implemented and coupled SEM and FM using a (2,6) FDTD interfacing scheme handling independent spatial and temporal resolutions. Coupling the Fourier method using four points per wavelength with the SEM using eight points per wavelength in a 5 m + 5m domain results in -36 dB interface errors and 9.1% relative errors compared to a reference solution. Using 12 points per wavelength for the SEM gives slightly better relative errors of 5.7%. The efficiency of the coupled method is compared against an SEM running in the entire domain. For a fixed domain size of 50 m, the efficiency of the coupled method is compared for different relative FM partition sizes, with an 18 times performance gain when 95% of the domain is running the FM. A more significant performance gain is expected to be achieved when going to larger 2D and 3D domains due to more degrees of freedom to be handled by the SEM. However, the workload at the interface will also grow compared to 1D, but we expect it to still be negligible compared to the saved computation time running the more expensive SEM solver.

Future work could consider the use of more accurate time-stepping schemes such as the Runge-Kutta methods, improving the numerical accuracy of the SEM method. A current limitation of the implemented method is the need for an additional SEM layer of first-order polynomials dominating the overall convergence rate.

REFERENCES

[1] N. H. Asmar. *Partial Differential equations with Fourier Series and Boundary Value Problems*. Pearson, Prentice Hall, 2 edition, 2005.

[2] C. De Boor. *A practical guide to splines*, volume 27. Springer, 2001.

[3] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2015. Algorithms, theory, and parallel implementation.

[4] A. P. Engsig-Karup, C. Eskilsson, and D. Bigoni. A stabilised nodal spectral element method for fully nonlinear water waves. *Journal of Computational Physics*, 318:1–21, 2016.

[5] B. Hamilton and S. Bilbao. FDTD Methods for 3-D Room Acoustics Simulation with High-Order Accuracy in Space and Time. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 25(11):2112–2124, 2017.

[6] J. A. Hargreaves, L. R. Rendell, and Y. W. Lam. A framework for auralization of boundary element method simulations including source and receiver directivity. *The Journal of the Acoustical Society of America*, 145(4):2625–2637, 2019.

[7] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods : algorithms, analysis, and applications*, volume 54. Springer, 2008.

[8] M. Hornikx, R. Waxler, and J. Forssén. The extended Fourier pseudospectral time-domain method for atmospheric sound propagation. *The Journal of the Acoustical Society of America*, 128(4):1632–1646, 2010.

[9] H. Kuttruff. *Room Acoustics*. CRC Press, 6 edition, 2016.

[10] R. Mehra, N. Raghuvanshi, L. Savioja, M. C. Lin, and D. Manocha. An efficient GPU-based time domain solver for the acoustic wave equation. *Applied Acoustics*, 73(2):83–94, 2012.

[11] A. Melander, E. Strøm, F. Pind, A. P. Engsig-Karup, C.-H. Jeong, T. Warburton, N. Chalmers, and J. S. Hesthaven. Massive parallel nodal discontinuous galerkin finite element method simulator for room acoustics. *Preprint on EPFL server*, 2020.

[12] T. Okuzono, T. Otsuru, R. Tomiku, and N. Okamoto. A finite-element method using dispersion reduced spline elements for room acoustics simulation. *Applied Acoustics*, 79:1–8, 2014.

[13] R. Pagán Muñoz and M. Hornikx. Hybrid Fourier pseudospectral/discontinuous Galerkin time-domain method for wave propagation. *Journal of Computational Physics*, 348:416–432, 2017.

[14] F. Pind, A. P. Engsig-Karup, C.-H. Jeong, J. S. Hesthaven, M. S. Mejling, and J. Strømann-Andersen. Time domain room acoustic simulations using the spectral element method. *The Journal of the Acoustical Society of America*, 145(6):3299–3310, 2019.

[15] N. Raghuvanshi, N. Galoppo, and M. C. Lin. Accelerated wave-based acoustics simulation. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 91–102. ACM, 2008.

[16] N. Raghuvanshi, R. Narain, and M. C. Lin. Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 15:789–801, September 2009.

[17] B. Van Genechten, O. Atak, B. Bergen, E. Deckers, S. Jonckheere, J. S. Lee, A. Maressa, K. Vergote, B. Pluymers, D. Vandepitte, and W. Desmet. An efficient Wave Based Method for solving Helmholtz problems in three-dimensional bounded domains. *Engineering Analysis with Boundary Elements*, 36(1):63–75, 2012.

[18] B. Van Hal, W. Desmet, D. Vandepitte, and P. Sas. Hybrid Finite Element - Wave Based Method for acoustic problems. *Computer Assisted Mechanics and Engineering Sciences*, 10(4):479–494, 2003.

[19] H. Xu, C. D. Cantwell, C. Monteserin, C. Eskilsson, A. P. Engsig-Karup, and S. J. Sherwin. Spectral/hp element methods: Recent developments, applications, and perspectives. *Journal of Hydrodynamics*, 30(1):1–22, 2018.

Report A

ACCELERATED SOUND PROPAGATION SIMULATIONS USING AN ERROR-FREE FOURIER METHOD COUPLED WITH THE SPECTRAL ELEMENT METHOD

Nikolas Borrel-Jensen
Acoustic Technology, Department of Electrical and Photonics Engineering
Technical University of Denmark, 2800 Kongens Lyngby, Denmark

August 9, 2023

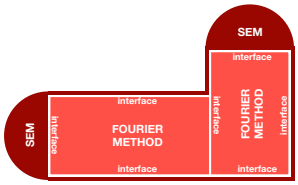
ABSTRACT

Simulating acoustics using numerical methods efficiently and accurately has been an active research area for the last decades and has applications in computer games, VR/AR, and architectural design. However, their extensive computation time makes these methods challenging for large scenes and broad frequency ranges. This work attempts to accelerate the simulations using rectangular decomposition, enabling error-free propagation in the bulk of the domain consisting of air. We exploit the analytical solution to the wave equation in rectangular domains calculated using the Fast Fourier Transform with near-optimal spatial discretization satisfying the Nyquist criterium. Coupling with the spectral element method near the boundaries results in a method capable of handling complex geometries with realistic boundaries, though with the caveat that additional errors and computational overhead may result from the interface. This paper will investigate the accuracy and efficiency of the proposed domain decomposition method compared to a spectral element implementation running in the entire domain.

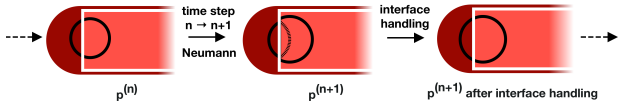
1 Introduction

Simulating acoustics in virtual spaces is an active research topic, with applications in computer games, VR/AR, and building acoustics. There are two types of methods for simulating acoustics: geometrical acoustics (GA) [23], where the sound propagation is simplified and modeled as rays; and numerical acoustics, solving the physical equation implicitly taking phenomena such as diffraction and scattering into account. The advantage of GA is its computational efficiency; however, it is only considered valid in the upper-frequency range. Also, diffraction and scattering are not directly taken into account and care must be taken to ensure smooth pressure transitions in interactive environments. On the contrary, numerical acoustics solve the underlying physics and take all physical phenomena into account. However, it comes with the cost of being computational demanding, especially when simulating broad frequency ranges and large domains. To overcome the demanding computations, we will develop an efficient and accurate domain decomposition method coupling an efficient and error-free Fourier method in rectangular domains with an SEM applied at the boundaries handling complex geometries and impedance boundaries.

The most used numerical methods for simulating acoustics are the finite element methods (FEM) [17], spectral element methods (SEM) [19], DG-FEM [16], finite-difference time-domain methods (FDTD) [3, 8, 13], boundary element methods [9, 24], the Wave-Based Method (WBM) [5, 25], and Pseudo-spectral Fourier methods [11, 12]. Domain decomposition methods (DDM) [1, 6] are widely used approaches, where the domain is split into many partitions. DDMs are often used for large-scale scientific applications to divide the problem into smaller parts distributed across multiple processors in parallel for efficiency purposes. In this work, we are not using domain decomposition in this regard, but as a matter to divide the domain into simpler partitions where specialized methods can be applied independently depending on the properties of the domain. Our method is inspired by Raghuvanshi et al. [15, 22], where an adaptive rectangular decomposition (ARD) method was proposed, exploiting the well-known analytical solution to the wave-equation in rectangular domains coupled with the FDTD method near the boundaries. The ARD



(a) Illustration of a domain decomposed into rectangular partitions running the error-free Fourier method coupled with the spectral element method near the boundaries.



(b) Time step from n to $n+1$ with Neumann boundary conditions at the interface for each of the methods running in the separate partitions. For the Fourier method, the time-stepping is done in the frequency domain by first transforming the pressure values to the frequency domain through the DFT. Then apply the FDTD interface handling to communicate sound pressures between the two partitions in the time domain. For the Fourier method, the updated values after time-stepping is transformed back to the time-domain by applying the iDCT.

Figure 1: Overview of the domain decomposition methods coupling the Fourier method in rectangular partitions with the spectral element method near the boundaries.

method has error-free propagation within the rectangular domains and can be computed very efficiently using the Fast Fourier Transform (FFT). Only numerical errors are introduced at the interfaces. We follow the same approach by coupling the Fourier method (FM) with the SEM (instead of the FDTD). The SEM is a very flexible method capable of handling complex geometries and impedance boundaries [19]. By coupling the efficient Fourier method running in rectangular domains, and the SEM near the boundaries, we can speed up the overall computation time for large scenes, where the ratio between air and boundaries is large. Related domain decomposition approaches have been taken to achieve similar goals. In [18], the time-domain Pseudo-Spectral element method was coupled with the DG-FEM method near the boundaries. In [20, 26, 27] the Wave-Based method is applied in convex domains and coupled with the second-order SEM in domains requiring geometrical flexibility.

Our method consists of three parts: 1) an error-free Fourier method (FM) running in rectangular domains, 2) the SEM for modeling complex geometries and realistic impedance boundaries, and 3) an FDTD scheme for interface handling. In Figure 1a the domain decomposition is illustrated.

2 Governing equations

The wave equation is

$$\frac{\partial^2 p(\mathbf{x},t)}{\partial t^2} - c^2\nabla^2 p(\mathbf{x},t) = F(\mathbf{x},t), \qquad t \in \mathbb{R}^+, \qquad \mathbf{x} \in \mathbb{R}^N. \tag{1}$$

where p is the pressure (Pa), t is the time (s) and c is the speed of sound in air (m/s) and $F(\mathbf{x},t)$ is a forcing function. An initial conditions can be used instead of a forcing function satisfied by using e.g. a Gaussian source for the pressure part and setting the velocity equal to zero

$$p(\mathbf{x}, t=0, \mathbf{x}_0) = \exp\left[-\left(\frac{\mathbf{x}-\mathbf{x}_0}{\sigma_0}\right)^2\right], \qquad \frac{\partial p(\mathbf{x}, t=0, \mathbf{x}_0)}{\partial t} = 0, \qquad \mathbf{x}_0 \in \mathbb{R}^N, \tag{2}$$

with σ_0 being the width of the pulse determining the frequencies to span and boundary conditions

$$\frac{\partial p(\mathbf{x},t)}{\partial \mathbf{n}} = \overline{v}, \qquad \mathbf{x} \in \Gamma_v, \tag{3}$$

$$p(\mathbf{x},t) = \overline{p}, \qquad \mathbf{x} \in \Gamma_p, \tag{4}$$

where $\overline{v}$ is the enforced velocity at the boundary Γ_v, $\overline{p}$ is the enforced pressure at the boundary Γ_p, and $\mathbf{n}$ is the normal pointing outwards from the boundary Γ. Equation (3) and (4) are the Neumann and Dirichlet boundary conditions, respectively, and the former is applied at the interface for coupling domains – as explained in Section 6 – with $\overline{v} = 0$. This condition corresponds to a 'hard wall' allowing displacement of the particles in the medium with the net directional force to be zero. Equation (3) is the Neumann boundary conditions and is applied at the interface for coupling domains – as explained in Section 6 – with $\overline{v} = 0$.

3 The Fourier method in rectangular domains

3.1 The analytical solution

It can be shown by using separation of variables [2] (p. 155-157), that any 1D pressure field $p(x,t)$ in *rectangular domains* with *Neumann boundary conditions* can be represented in the form of a general series representation

$$p(x,t) = \sum_{i=0}^{N-1} M_i(t)\Phi_i(x), \qquad \text{where} \quad M_i(t) = ae^{jck_it} + be^{-jck_it}, \quad k_i = \pi\frac{i}{l}, \tag{5}$$

and i is the mode, N is the maximum mode to include depending on the required frequency range, $M_i(t)$ is the time-varying mode coefficients [14, 21], $j = \sqrt{-1}$ is the complex number, k_i is the wavenumber, and $\Phi_i(x) = \cos(k_ix)$ are the eigenfunctions of the Laplacian for a rectangular domain. The time constants a and b depend on the initial conditions.

3.2 The discrete formulation

We can interpret the analytical solution (5) on a discrete uniform grid x_i with the highest wavenumber being spatially sampled at the Nyquist rate. Assuming that the signal is properly band-limited and that enough modes N are included to capture the band-limited signal, the discretization introduces no numerical errors.

In the discrete interpretation, Equation (5) is similar to the inverse Discrete Cosine Transform, with ϕ_i being the cosine basis vectors. Hence, we can convert from modal coefficients $\mathbf{M}$ to pressure values $\mathbf{p}$ as

$$\mathbf{p}(t) = \text{iDCT}(\mathbf{M}(t)), \tag{6}$$

where $\mathbf{p}$ is the $N \times 1$ pressure vector and $\mathbf{M}(t)$ is the $N \times 1$ vector including all modes for time t. Reinterpreting the wave equation $\frac{\partial^2 p}{\partial t^2} - c^2\frac{\partial^2 p}{\partial x^2} = F$ in a discrete setting for the spatial dimensions and taking the cosine transform of both sides of the equality sign yields

$$\frac{\partial^2}{\partial t^2}\mathbf{M}(t) + c^2\mathbf{k}^2 \odot \mathbf{M}(t) = \text{DCT}(\mathbf{F}(t)), \tag{7}$$

where $\mathbf{F}(t)$ is a $N \times 1$ vector with pressure values of the forcing function in all spatial nodes at time t, $\mathbf{M}(t)$ is a $N \times 1$ vector with mode coefficients corresponding to time t, $\mathbf{k}$ is a $N \times 1$ vector including the wave numbers, and $\odot$ is the Hadamard product operator. Disregarding the forcing term, the above equation describes a set of independent simple harmonic oscillators, each vibrating with its characteristic frequency $\omega_i = ck_i$.

In the case where the forcing term is a constant transformed into mode-space $\tilde{F}$, we can simply solve the equation $\frac{\partial^2}{\partial t^2}M_i + \omega_i^2 M_i - \tilde{F}_i = 0$, but since the forcing term changes with time, we need to derive a temporal update scheme for this. We will assume that the forcing term is constant over a time-step Δt, which is satisfied when proper Nyquist sampling is applied. The forcing term may be transformed into mode-space as $\tilde{\mathbf{F}}(n\Delta t) \triangleq \text{DCT}(\mathbf{F}(n\Delta t))$. We will use the second-order centered difference multiplied by the term $\frac{\omega^2\Delta t^2}{2(1-\cos(\omega\Delta t))}$ originating from the solution to the simple harmonic oscillator, obtaining the update scheme [22]

$$M_i^{(n+1)} = 2M_i^{(n)}\cos(\omega_i\Delta t) - M_i^{(n-1)} + \frac{2\tilde{F}^{(n)}}{\omega_i^2}(1 - \cos(\omega_i\Delta t)). \tag{8}$$

Arbitrary source signals can be included as forcing functions F from Equation 7. Alternatively, source signals can be included as initial conditions (typically Gaussian functions). Since the wave-equation is a second-order in time, two initial conditions are needed in both cases.

The simulation loop for solving the wave equation in a rectangular domain is summarized here:

- Initialization ($n = 0$ and $n = 1$):
  1. Transform a forcing signal $\mathbf{F}^{(0)}$ and $\mathbf{F}^{(-1)}$ into mode-space $\tilde{\mathbf{F}}^{(0)}$ and $\tilde{\mathbf{F}}^{(-1)}$, respectively, by using the DCT.
  2. $\mathbf{M}^{(0)}$ and $\mathbf{M}^{(-1)}$ are initialized in either ways:

     Source injection. Initial conditions is trivially set to zero as $\mathbf{M}^{(0)} = 0$ and $\mathbf{M}^{(-1)} = 0$.

     Source as initial conditions. When the initial condition is $\mathbf{M}^{(0)} = \tilde{\mathbf{F}}^{(0)}$, then the second condition is chosen such that the function is correctly evaluated at the previous time-step $\mathbf{M}^{(-1)} = \tilde{\mathbf{F}}^{(-1)}$.

- Simulation loop ($n \geq 2$):
  1. Calculate all modes ($i = 0 \dots N-1$) for time-step $n+1$ denoted by the matrix $\mathbf{M}^{(n+1)}$ using Equation (8).
  2. Transform all modes $\mathbf{M}^{(n+1)}$ to pressure values by applying iDCT.
  3. Transform the forcing signal $\mathbf{F}^{(n+1)}$ into mode-space $\bar{\mathbf{F}}^{(n+1)}$ by using the DCT for the next time-step $n+1$.
  4. Set $\mathbf{M}^{(n-1)} = \mathbf{M}^{(n)}$, $\mathbf{M}^{(n)} = \mathbf{M}^{(n+1)}$ and $\mathbf{F}^{(n)} = \mathbf{F}^{(n+1)}$ in Equation (8).

3.3 Discrete Fourier Transform for imposing Neumann boundary conditions

The formulation in the analytical formulation in (5) is real and hence, to ensure a real-valued frequency spectrum after applying the Discrete Fourier Transform, an even Fourier continuation $p_k = p_{-k}$ and periodicity[1] are required. Mirroring the samples around a boundary satifies the zero derivative Neumann boundary condition in that point. The extended time signal $\hat{p}_k$ of lenght $(2N-2)$ is then

$$\hat{p}_k \quad = \quad \begin{cases} p_k, & \text{for} \quad 0 \leq \quad k < N, \\ p_{2N-k}, & \text{for} \quad N \leq \quad k < 2N-1. \end{cases} \tag{9}$$

The single-sided frequency spectrum M_i consisting of N bins is used for time-stepping in Equation (8). Again, but now in the Fourier domain, the even and periodic double-sided spectrum $\hat{M}_i$ of size $(2N-2)$ is reconstructed and the inverse Fourier Transform is applied to retrieve the time pressures back as

$$\hat{M}_i \quad = \quad \begin{cases} M_i, & \text{for} \quad 0 \leq \quad i < N, \\ M_{2N-i}, & \text{for} \quad N \leq \quad i \leq 2N-1. \end{cases} \tag{10}$$

Extracting the first N pressures values from $\mathcal{F}^{-1}(\hat{M}_i)$ gives us the final result. Hence, the DCT and iDCT used in Section 3 and the remaining sections are referring to the definitions below

$$\begin{aligned} \text{DCT}\left[p_k\right] &= \text{Re}\left(\mathcal{F}[\hat{p}_k]\right), \\ \text{iDCT}[M_i] &= \text{Re}\left(\mathcal{F}^{-1}[\hat{M}_i]\right). \end{aligned} \tag{11}$$

3.4 Relation between normal modes and spatial dimension

The spatial resolution is dependent on the maximum simulation frequency determined in this setup by the number of modes and the spatial dimension. We have the following relations between wavelength $\lambda_{\min}$, domain size l, number of modes N and max frequency $f_{\max}$

$$\lambda_{\min} \quad = \quad \frac{l}{N} \tag{12}$$

$$f_{\max} \quad = \quad \frac{c}{\lambda_{\min}} \tag{13}$$

The grid resolution following the Nyquist Theorem is

$$\Delta x \quad = \quad \frac{\lambda_{\min}}{2} = \frac{l}{2N}$$

The relation between the time domain dimension K and the frequency domain dimension N is

$$K = \frac{l}{\Delta x} = 2N \quad \Rightarrow \quad N = K/2$$

Hence, $2N$ modes should be included to match the spatial resolution Δx with K nodes. From DSP theory, we know that transforming a time signal s with K number of samples using the Fourier transform results in a double-sided frequency signal S with K bins. Though, since we are performing a Fourier extension as explained in Section 3.3, the remaining K bins include only the single-sided frequency signal containing all necessary information to reconstruct the time signal.

[1] if we have a signal 'abcd', then the even extension would be 'abcdcb'.

4 Finite-Difference Time-Domain scheme

4.1 Update scheme

The second-order centered finite-difference (leapfrog) scheme is used for time-integration

$$\frac{\partial^2 p}{\partial t^2} = \frac{p^{(n+1)} - 2p^{(n)} + p^{(n-1)}}{\Delta t^2} + \mathcal{O}(\Delta t^2), \tag{14}$$

where n is the time-step, Δt is the discretization resolution and $\mathcal{O}(\Delta t^2)$ is the truncation error. The sixth-order centered finite-difference scheme is used for spatial differentiation

$$\frac{\partial^2 p}{\partial x^2} = \frac{2p_{i-3} - 27p_{i-2} + 270p_{i-1} - 490p_i + 270p_{i+1} - 27p_{i+2} + 2p_{i+3}}{\Delta x^2} + \mathcal{O}(\Delta x^6), \tag{15}$$

where Δx is the spatial resolution. The (2,6) FDTD update scheme can then be summarized

$$p_i^{(n+1)} = 2p_i^{(n)} - p_i^{(n-1)} + c^2\Delta t^2 \times \frac{2p_{i-3}^{(n)} - 27p_{i-2}^{(n)} + 270p_{i-1}^{(n)} - 490p_i^{(n)} + 270p_{i+1}^{(n)} - 27p_{i+2}^{(n)} + 2p_{i+3}^{(n)}}{\Delta x^2} \tag{16}$$

The relation between the spatial and temporal resolution is

$$\Delta t = \frac{\Delta x}{\text{CFL} \times c}, \tag{17}$$

where CFL is the Courant condition depending on the scheme. Unless other is stated, we will use CFL $= \sqrt{3}$ in this work.

5 Spectral element method

5.1 Update scheme

We present the wave equation from Equation (1) in 2D with velocity (Neumann) boundary condition on Γ_v from Equation (3), and pressure (Dirichlet) boundary conditions on Γ_p from Equation (4). The weak form is achieved by multiplying with a test function ϕ vanishing at the endpoints and integrating over the domain as

$$\int_\Omega \left(\frac{\partial^2 p}{\partial t^2} - c^2\nabla^2 p\right)\phi \mathrm{d}\Omega - c^2\left[\oint_{\Gamma_v} (\mathbf{n}^T\nabla p - \overline{\mathbf{v}}_n)\phi \mathrm{d}\Gamma_v + \oint_{\Gamma_p} (p - \overline{p})\phi \mathrm{d}\Gamma_p\right] = \int\int_\Omega F\phi \mathrm{d}x\mathrm{d}y, \tag{18}$$

where $\mathbf{n}$ is the surface normal vector pointing outwards, $\overline{\mathbf{v}}_n$ and $\overline{\mathbf{p}}$ are the enforced velocity and pressure at the boundaries, respectively. By applying integration by parts, it is possible to reduce the derivation order by one for the Laplacian terms. Regarding the integration term, Green's first identity

$$\int_\Omega (\psi\nabla^2\varphi + \nabla\psi\nabla\varphi)\mathrm{d}\Omega = \oint_\Gamma \psi\left(\frac{\partial\varphi}{\partial\mathbf{n}}\right)\mathrm{d}\Gamma \tag{19}$$

can be applied to eliminate the normal pressure derivatives (Green's identity can also be seen as the equivalent to integration by parts in higher dimensions), thus

$$\int_\Omega \frac{\partial^2 p}{\partial t^2}\phi \mathrm{d}\Omega + \int_\Omega c^2\nabla p\nabla\phi + c^2\left[\oint_{\Gamma_v} \overline{\mathbf{v}}_n\phi \mathrm{d}\Gamma_v + \oint_{\Gamma_p} (p - \overline{p})\phi \mathrm{d}\Gamma_p\right] = \int\int_\Omega F\phi \mathrm{d}x\mathrm{d}y, \tag{20}$$

Now, a truncated series expansion for the unknown variable p is introduced

$$p(x, y) \approx \hat{p}(x, y) = \sum_{i=1}^{K} \hat{p}_i N_i(x, y), \tag{21}$$

where $N_i(x,y) \in V$ is chosen from the space of globally continous piecewise orthogonal polynomials of degree at most P defined as $V = \{\mathbb{P}^P\}$ possessing the cardinal property $N_i(\mathbf{x}_j) = \delta_{ij}$. In this work, Lagrange polynomials are chosen spanning the space V. Insert Equation (21) for p into Equation (20) and setting the test function equal to each of the basis functions $\phi = \{N_i(x,y)\}_i^K$, with K being the number of nodes, we get the following semi-discrete system

$$\mathcal{M}\frac{\partial^2 \mathbf{p}}{\partial t^2} = -c^2\mathcal{S}\mathbf{p} + \mathcal{M}\mathbf{f} \tag{22}$$

where $\mathcal{M}$ and $\mathcal{S} = \mathcal{S}_x + \mathcal{S}_y$ are the so-called mass and stiffness matrices, respectively, and where the terms $\mathbf{v}_n\mathcal{B}_v + \mathcal{B}_p\mathbf{p}$ concerning the boundary conditions, are included in the stiffness matrix. In Section 5.2 and 5.3, these matrices are defined in more details. The update formula after applying the 2nd order centered finite difference scheme in time from Equation (14) yields

$$\mathbf{p}^{(n+1)} = 2\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)} - c^2\Delta t^2\mathcal{M}^{-1}\left(\mathcal{S}\mathbf{p}^{(n)}\right) + \Delta t^2\mathbf{f}^{(n)} \tag{23}$$

5.2 Defining the mass and stiffness matrices

The mass and stiffness matrices $\mathcal{M}$ and $\mathcal{S}$ are defined by first introduced the following *global* matrices from the global piecewise basis functions N_i

$$\begin{aligned}
\mathcal{M}_{ij} &= \int\int_\Omega N_i(x,y)N_j(x,y)\mathrm{d}x\mathrm{d}y, \\
\mathcal{S}_{x,ij} &= \int\int_\Omega \frac{\partial N_i(x,y)}{\partial x}\frac{\partial N_j(x,y)}{\partial x}\mathrm{d}x\mathrm{d}y, \\
\mathcal{S}_{y,ij} &= \int\int_\Omega \frac{\partial N_i(x,y)}{\partial y}\frac{\partial N_j(x,y)}{\partial y}\mathrm{d}x\mathrm{d}y \\
\mathcal{B}_{ij} &= \int\int_\Gamma N_i(x,y)N_j(x,y)\mathrm{d}x\mathrm{d}y,
\end{aligned} \tag{24}$$

It is convenient to introduce the concept of a *local element matrix*; due to the nature of the global piecewise basis function in Equation (24), the integrals are only non-zero in the overlapping regions where the nodes i, j belong to the same element. We therefore define the local element matrices, where (i,j) are the indexes of the q'th non-overlapping element e_q (matrix $\mathcal{B}$ omitted)

$$\begin{aligned}
\mathcal{M}_{ij}^{(q)} &= \int\int_{e_n} N_i^{(q)}(x,y)N_j^{(q)}(x,y)\mathrm{d}x\mathrm{d}y, \\
\mathcal{S}_{x,ij}^{(q)} &= \int\int_{e_n} \frac{\partial N_i^{(q)}(x,y)}{\partial x}\frac{\partial N_j^{(q)}(x,y)}{\partial x}\mathrm{d}x\mathrm{d}y, \\
\mathcal{S}_{y,ij}^{(q)} &= \int\int_{e_n} \frac{\partial N_i^{(q)}(x,y)}{\partial y}\frac{\partial N_j^{(q)}(x,y)}{\partial y}\mathrm{d}x\mathrm{d}y,
\end{aligned} \tag{25}$$

The global matrices $\mathcal{M}$, $\mathcal{S}$, and $\mathcal{B}$ from Equation (24) can be constructed from the local matrices in Equation (25) by summing the (local) element contributions

$$\mathcal{M}_{ij} = \int\int_{e_q} N_iN_j\mathrm{d}x\mathrm{d}y = \sum_{n=1}^{N_{\text{el}}}\int\int_{e_q} N_i^{(q)}N_j^{(q)}\mathrm{d}x\mathrm{d}y, \tag{26}$$

where N_{el} are the number of elements, and similarly for the matrices $\mathcal{S}$ and $\mathcal{B}$. Note, that the resulting matrix is sparse, due to the local support of the basis functions, allowing for using efficient solvers for the system of equations.

5.3 Solving the integrals

The integrations in (25) can be calculated exactly without resorting to, e.g. Gaussian quadratures. It is convenient to introduce a reference triangle element [7]

$$\mathcal{I} = \{(r,s)|(r,s) \geq -1 \leq r,s; r+s \leq 0\}$$

The nodal Lagrange basis functions $N_n(r,s)$ can be determined on the reference element $\mathcal{I}$ using the Vandermonde matrix $\mathcal{V}$

$$N_n(r,s) \quad = \quad \sum_{n=1}^{P+1} \left(\mathcal{V}^T\right)^{-1}_{i,n} \psi_n(r,s) \tag{27}$$

where $\psi_n(r,s)$ are modal Legendre polynomials and the nodal distribution of the collocation points r,s of the reference element is of the Legendre-Gauss-Lobatto (LGL) kind. Inserting the above into the expression for the mass matrix in (25), but on the reference element yields

$$\mathcal{M} = (\mathcal{V}\mathcal{V}^T)^{-1} \tag{28}$$

where $\mathcal{V}$ is the Vandermonde matrix. The mapping from the local reference element $\mathcal{I}$ to the global element e_q on (x,y) in (25) is

$$\begin{aligned} \mathcal{M}^{(q)}_{ij} = \int\int_{e_n} N^{(q)}_i(x,y) N^{(q)}_j(x,y) \mathrm{d}x\mathrm{d}y = \\ \mathcal{J}^{(q)} \int\int_{\mathcal{I}} N^{(q)}_i(r,s) N^{(q)}_j(r,s) dr ds \end{aligned} \tag{29}$$

where $\mathcal{J}^{(q)}$ is the Jacobian mapping from local to the global element q as $(x,y) \rightarrow (r,s)$. Regarding the stiffness matrix, we use

$$\frac{\partial}{\partial r} N_i(r,s) = \sum_{n=1}^{P+1} \frac{\partial}{\partial r} N_i(r_n, s_n) N_n(r,s) \tag{30}$$

and inserting the above term into the expression for the stiffness matrix, but (again) on the reference element yields

$$\mathcal{S}_r = \mathcal{D}_r^T \mathcal{M} \mathcal{D}_r, \quad \mathcal{S}_s = \mathcal{D}_s^T \mathcal{M} \mathcal{D}_s \tag{31}$$

where $\mathcal{D}_r$ is the differentiation matrix

$$\mathcal{D}_r \quad = \quad \mathcal{V}_r \mathcal{V}^{-1} \tag{32}$$

The mapping from the reference element to the global element q for the stiffness matrix is again done using the Jacobian $\mathcal{J}^{(q)}$

$$\mathcal{S}^{(q)} = \mathcal{J}^{(q)} \mathcal{D}_x^T \mathcal{M} \mathcal{D}_x + \mathcal{J}^{(q)} \mathcal{D}_y^T \mathcal{M} \mathcal{D}_y \tag{33}$$

where

$$\mathcal{D}_x = r_x \mathcal{D}_r + s_x \mathcal{D}_s, \quad \mathcal{D}_y = r_y \mathcal{D}_r + s_y \mathcal{D}_s \tag{34}$$

and r_x, r_y, s_x, s_y are geometrical factors. To summarize, the mass and stiffness matrices for the q'th element can be calculated as

$$\mathcal{M}^{(q)} = \mathcal{J}^{(q)} (\mathcal{V}\mathcal{V}^T)^{-1}, \quad \mathcal{S}^{(q)} = \mathcal{J}^{(q)} \mathcal{D}_x^T \mathcal{M} \mathcal{D}_x \tag{35}$$

6 Interface handling

We will derive the interface scheme for communicating pressure values between partitions following the same approach as Raghuvanshi [21, 22] to couple the FM in rectangular partitions with the SEM near the boundaries. For clarity of presentation, we will consider the 1D wave equation using the second-order centered differences. The wave equation (1) discretized in space can be written as $\frac{\partial^2 P}{\partial t^2} - KP = F(t)$, where K represents the Laplacian operator at each node as a Discrete Laplacian Matrix and F is the forcing term at each node. As a result of spatial discretization, K is transformed into a symmetric matrix

$$K = \frac{c^2}{\Delta x^2} \begin{bmatrix} \ddots & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & -2 & 1 & & \\ & & & 1 & -2 & 1 & \\ & & & & & & \ddots \end{bmatrix}. \tag{36}$$

Assume that the grid consists of N nodes. Care must be taken at the boundary, since calculating the pressure value p_N at the right-most node N would require the neighboring pressure value p_{N+1} located outside the grid for a Neumann boundary condition imposed by mirroring the pressure half-way between the nodes or at the boundary node. We will denote the pressure values at these ghost nodes as the *residual*. Eliminating the ghost nodes happens by discretizing the BCs and inserting these expressions into the scheme. For example, for a right boundary node x_N, the residual is added to the inner partition as

$$\begin{aligned} p''_{N+1/2}(x_N) &= \frac{p(x_{N-1}) - 2p(x_N) + p(x_{N+1})}{\Delta x^2} \overset{\text{Neumann}}{\underset{\text{N+1/2}}{=}} \frac{p(x_{N-1}) - p(x_N)}{\Delta x^2}, \\ p''_{N}(x_N) &= \frac{p(x_{N-1}) - 2p(x_N) + p(x_{N+1})}{\Delta x^2} \overset{\text{Neumann}}{\underset{\text{N}}{=}} \frac{2p(x_{N-1}) - 2p(x_N)}{\Delta x^2}. \end{aligned} \tag{37}$$

These observations can be used to formulate the interface handling between two separate partitions. Suppose we wish to split the domain into two equal partitions with N nodes each, such that each might be treated independently. P is a vector of length $2N$. This can be done by decoupling K into a block diagonal form while accounting properly for the off-diagonal entries $K = A + C$, where the decoupled matrix A and the coupling matrix C are given by (interface located at $N + 1/2$)

$$A = \frac{c^2}{\Delta x^2} \left[\begin{array}{ccc|ccc} \ddots & & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -1 & & & \\ \hline & & & -1 & 1 & \\ & & & 1 & -2 & 1 \\ & & & & & \ddots \end{array}\right], \qquad C = \frac{c^2}{\Delta x^2} \left[\begin{array}{ccc|ccc} 0 & & & & & \\ & \ddots & & & & \\ & & -1 & 1 & & \\ \hline & & 1 & -1 & & \\ & & & & \ddots & \\ & & & & & 0 \end{array}\right]. \tag{38}$$

The pressure is first updated using A, and the residual part not taken into account is then computed using C. The coupling matrix C can intuitively be seen as a communication step to pass pressures between two partitions, and the action of C is to compute the gradient of the pressure at the interface (scaled by some factor). Rewriting in terms of the decoupled and coupled matrices yields $\frac{\partial^2 P}{\partial t^2} - AP = F(t) + CP$, where the coupling term CP can be seen as an additional source term accounting for the alignment of pressures between partitions as $\hat{F}(t) = F(t) + CP$.

We can formulate a unified framework for communicating pressures between partitions running *any* method. The boundary condition will be imposed at the pressure node N (not $N + 1/2$ as above) since this will ease the implementation for the SEM where boundaries are explicitly defined at the nodal points. Without lack of generality, assume that two independent (2,2) FDTD update schemes are running in each partition and denote the pressures in partition 1 as $p_{1,i}$ and partition 2 as $p_{2,i}$ with subscripts denoting the partition and corresponding node index, respectively. Then, the interface communication can be handled as follows:

1. Calculate the pressures in each domain as completely independent partitions with Neumann boundary condition at the interface

$$\begin{aligned} p_{1,N}^{(n+1)} &\leftarrow \frac{c^2 \Delta t_1^2}{\Delta x_1^2} \left(2p_{1,N-1}^{(n)} - 2p_{1,N}^{(n)}\right) + 2p_{1,N}^{(n)} - p_{1,N}^{(n-1)} + F_{1,N}^{(n)}, \\ p_{2,1}^{(n+1)} &\leftarrow \frac{c^2 \Delta t_2^2}{\Delta x_2^2} \left(-2p_{2,1}^{(n)} + 2p_{2,2}^{(n)}\right) + 2p_{2,1}^{(n)} - p_{2,1}^{(n-1)} + F_{2,1}^{(n)}. \end{aligned} \tag{39}$$

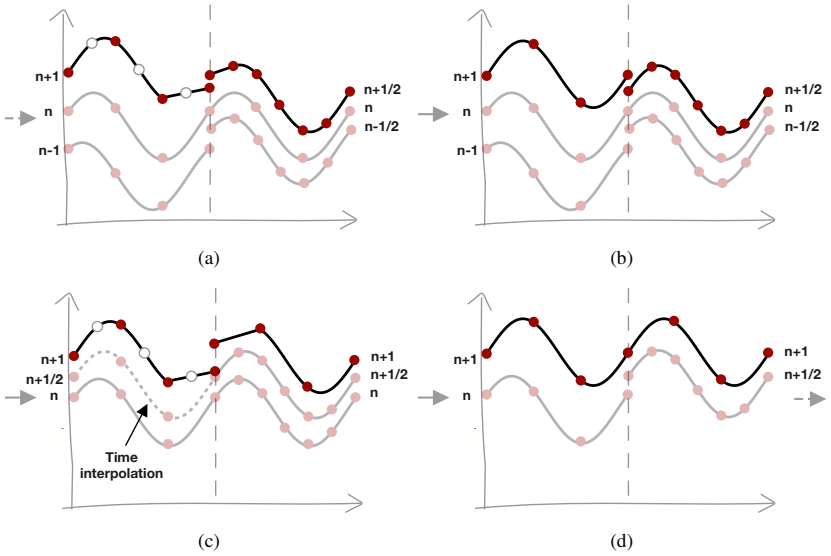
2. Remove the residual part at time n corresponding to the reflected pressures

$$p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} - \frac{c^2 \Delta t_1^2}{\Delta x_1^2} \left(p_{1,N-1}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} - \frac{c^2 \Delta t_2^2}{\Delta x_2^2} \left(p_{2,2}^{(n)}\right). \tag{40}$$

3. Transfer the removed residual part at time n to the neighboring partition(s)

$$p_{1,N}^{(n+1)} \leftarrow p_{1,N}^{(n+1)} + \frac{c^2 \Delta t_1^2}{\Delta x_1^2} \cdot \left(p_{2,2}^{(n)}\right), \qquad p_{2,1}^{(n+1)} \leftarrow p_{2,1}^{(n+1)} + \frac{c^2 \Delta t_2^2}{\Delta x_2^2} \left(p_{1,N-1}^{(n)}\right). \tag{41}$$

Figure 2: Interface handling with twice refined spatial and temporal resolutions in the right partition compared to the left partition. (a) Neumann reflections present at time $n+1$ and $n+1/2$ with spatial interpolation in partition 1. (b) Interface handling for time $n+1$ and $n+1/2$ by using the corresponding Δx_1, Δt_1, Δx_2, Δt_2 values for the FDTD scheme in Equation (39)-(41). (c) Temporal interpolation in the coarse domain for calculating $n+1$. (d) Interface handling for the fine domain (only) at time $n+1$.



Separating the pressure update scheme inside the cavity from the interface handling scheme makes it clear that we can transfer pressures between partitions running any schemes. The sixth-order update scheme used in this work can be derived in a similar manner. The spatial and temporal resolutions will differ between partitions if the efficiency of the Fourier method is to be exploited and therefore piece-wise cubic Hermite interpolation is used in both space and time [4]. The procedure is depicted in Figure 3.

6.1 6th order Laplacian

The coupling between domains using the sixth-order finite-difference scheme follows the same procedure as the second-order finite-difference scheme. We will write up the matrices and illustrate the coupling graphically. Again, Neumann boundary condition implies even symmetry of the pressure field half-way between the interface node $N+1/2$, i.e. assume that node N and $N+1$ lies inside two different domain, then $p_{1,N} = p_{2,1}$, $p_{1,N-1} = p_{2,2}$ and $p_{1,N-2} = p_{2,3}$ summarized in the Discrete Laplacian Matrix K

$$K = \frac{c^2}{\Delta x^2} \left[\begin{array}{cccccc|cccccc} \ddots & & & & & & & & & & & \\ 2 & -27 & 270 & -490 & 270 & -27 & 2 & & & & & \\ & 2 & -27 & 270 & -490 & 270 & -27 & 2 & & & & \\ & & 2 & -27 & 270 & -490 & 270 & -27 & 2 & & & \\ \hline & & & 2 & -27 & 270 & -490 & 270 & -27 & 2 & & \\ & & & & 2 & -27 & 270 & -490 & 270 & -27 & 2 & \\ & & & & & 2 & -27 & 270 & -490 & 270 & -27 & 2 \\ & & & & & & & & & & \ddots & \end{array}\right]. \tag{42}$$

Removing the residual part of the stencil (ghost points outside the domains) and re-adding these inside the domain gives

$$A = \frac{c^2}{\Delta x^2}\left[\begin{array}{cccccc|cccccc} \ddots & & & & & & & & & & & \\ 2 & -27 & 270 & -490 & 270 & -25 & 0 & & & & & \\ & 2 & -27 & 270 & -488 & 243 & 0 & 0 & & & & \\ & & 2 & -25 & 243 & -220 & 0 & 0 & 0 & & & \\ \hline & & & 0 & 0 & 0 & -220 & 243 & -25 & 2 & & \\ & & & & 0 & 0 & 243 & -488 & 270 & -27 & 2 & \\ & & & & & 0 & -25 & 270 & -490 & 270 & -27 & 2 \\ & & & & & & & & & & & \ddots \end{array}\right], \tag{43}$$

and the coupling matrix, C is given by

$$C = \frac{c^2}{\Delta x^2}\left[\begin{array}{cccccc|ccccc} 0 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & & & -2 & 2 & & & & & \\ & & & -2 & 27 & -27 & 2 & & & & \\ & & -2 & 27 & -270 & 270 & -27 & 2 & & & \\ \hline & & 2 & -27 & 270 & -270 & 27 & -2 & & & \\ & & & 2 & -27 & 27 & -2 & & & & \\ & & & & 2 & -2 & & & & & \\ & & & & & & & & \ddots & & \\ & & & & & & & & & 0 & \end{array}\right].$$

The matrix A is now enforcing Neumann boundaries around the interface $p_{N+1/2}$ assuming p_N is at the interface boundary in the original formulation using K. The framework for the sixth-order scheme is similar to the steps 1-3 for the 2nd-order scheme, and the corresponding steps 2-3 collapsed into one step for $p_{1,N}^{(n+1)}$ is:

$$\begin{aligned} p_{1,N}^{(n+1)} \leftarrow\ & p_{1,N}^{(n+1)} \\ & - \frac{c^2\Delta t_1^2}{180\Delta x_1^2}\cdot\left(270p_{1,N-1}^{(n)} - 27p_{1,N-2}^{(n)} + 2p_{1,N-3}^{(n)}\right) \\ & + \frac{c^2\Delta t_2^2}{180\Delta x_2^2}\cdot\left(270p_{2,2}^{(n)} - 27p_{2,3}^{(n)} + 2p_{2,4}^{(n)}\right). \end{aligned} \tag{44}$$

The remaining pressure values $p_{1,N-1}$ and $p_{1,N-2}$ near the interface can be calculated in a similar manner.

6.2 Handling different spatial and temporal resolutions in domains

The spatial and temporal resolutions will differ between partitions if the efficiency of the Fourier method is to be exploited. Therefore, interpolation is needed in time and space as depicted in Figure 3. The procedure is outlined in the following, where the time scale is relative to the coarse (left) domain. Piece-wise cubic Hermite interpolation is used[2] in both space and time, choosing slopes such that the second derivative is continuous [4].

(a) Calculate pressure values $p_1^{(n+1)}$ and $p_2^{(n+1/2)}$ with individual temporal and spatial resolutions.

(b) Do interface handling with individual temporal resolutions Δt_1 and Δt_2:

Partition 1 adjustment. Do spatial interpolation in partition 2 with Δx_1 and apply interface handling as outlined in Eq. (44) with $\Delta x_2 \leftarrow \Delta x_1$ and $\Delta t_1 \neq \Delta t_2$.

Partition 2 adjustment. Do spatial interpolation in partition 1 with Δx_2 and apply interface handling with $\Delta x_1 \leftarrow \Delta x_2$ and $\Delta t_1 \neq \Delta t_2$.

(c) Calculate pressure values $p_1^{(n+1)}$ and $p_2^{(n+1)}$ using fine Δt_2 temporal resolution in both partitions $\Delta t_1 \leftarrow \Delta t_2$:

Temporal interpolation. Obtain $p_1^{(n+1/2)}$ by interpolating in time using $p_1^{(n)}$ and $p_1^{(n+1)}$.

[2] the Matlab 2021b function 'spline' has been used.

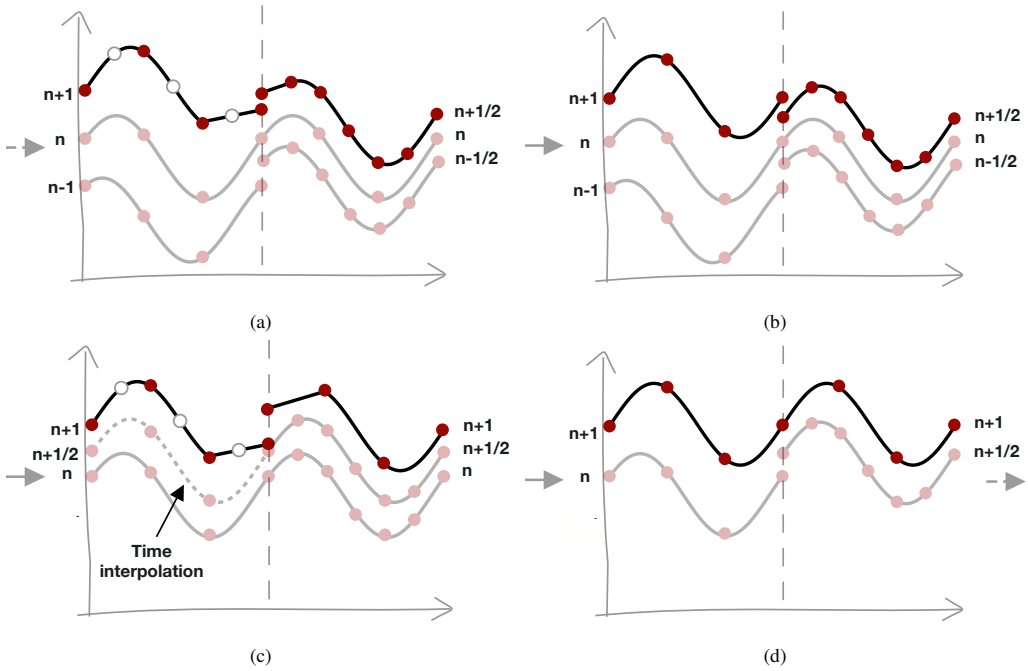


Figure 3: Illustration of the interface handling states with twice refined spatial and temporal resolutions in the right partition compared to the left partition. (a) State where Neumann reflections are present at time $n+1$ and $n+1/2$ for the coarse and fine resolution partitions, respectively. Spatial interpolation is done in partition 1, matching partition 2. (b) Interface handling compensating for the reflections directly between pressures corresponding to time $n+1$ and $n+1/2$. The reflected pressures are correctly being compensated for by using the corresponding Δx_1, Δt_1, Δx_2, Δt_2 values for the FD update scheme in Eq. (44). (c) To calculate $n+1$ for the fine resolution partition (right), temporal interpolation is done in the coarse domain (left) to achieve $n+1$ needed for interface handling. The pressures at $n+1$ in (a) could also have been used, but fewer interface errors are introduced in this manner. (d) Interface handling is done using the pressure values at time $n+1$. The pressure values at $n+1/2$ for the coarse domain are then discarded.

Time n+1 pressures calculations. $p_1^{(n+1)}$ is calculated from $p_1^{(n)}$ and interpolated $p_1^{(n+1/2)}$ setting $\Delta t_1 \leftarrow \Delta t_2$; $p_2^{(n+1)}$ is calculated using $p_2^{(n)}$ and (interface corrected) $p_2^{(n+1/2)}$.

(d) Do interface handling ($\Delta t_1 \leftarrow \Delta t_2$)

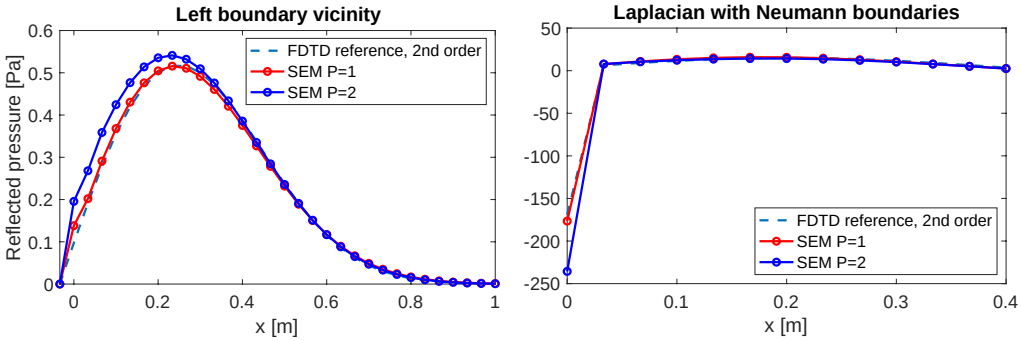
Partition 1 adjustment. Do spatial interpolation in partition 2 with Δx_1 and apply interface handling with $\Delta x_2 \leftarrow \Delta x_1$ and $\Delta t_1 \leftarrow \Delta t_2$.

Partition 2 adjustment. Do spatial interpolation in partition 1 with Δx_2 and apply interface handling with $\Delta x_1 \leftarrow \Delta x_2$ and Δt_2.

Note that interpolation in time is done *after* adjusting for reflections caused by the Neumann boundary condition; this is important since the interpolation would otherwise be unprecise, including significant errors for the interface handling at time $n+1/2$.

6.3 Interfacing with the Spectral-Element method

Since the interface handling is independent of the methods running in the partitions, coupling the FM with the SEM can be implemented. However, due to the non-physical behavior caused by the Neumann condition at the interface at each time step before compensating, non-smooth second derivatives are introduced. This type of behavior is known as 'shocks' in the literature, and it is well-known to introduce challenges for the SEM. An investigation has been made



(a) The left-travelling wave reflecting at a left Neumann boundary when subtracting the interface residual at each time step. We notice that much bigger errors for higher-order SEM compared to the first-order SEM and the FDTD method.

(b) The Laplacian term imposing Neumann boundaries introduces a shock, causing the second-order derivatives to be non-smooth when subtracting the interface residual at each time step. We notice that much bigger errors for higher-order SEM compared to the first-order SEM and the FDTD method.

Figure 4: Accuracy of the Laplacians for the FDTD scheme and SEM when the interface residual is subtracted.

by comparing the Laplacian term from Eq. (1) for the SEM and FDTD methods

$$\text{(SEM)} \quad \mathbf{p}^{(n+1)} = 2\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)} \overbrace{\boxed{-c^2\Delta t^2 \mathcal{M}^{-1}\left(\mathcal{S}\mathbf{p}^{(n)}\right)}}^{\text{SEM Laplacian}} + \Delta t^2 \mathbf{f}^{(n)}, \tag{45}$$

$$\text{(FDTD)} \quad \mathbf{p}^{(n+1)} = 2\mathbf{p}^{(n)} - \mathbf{p}^{(n-1)} \overbrace{\boxed{+\frac{c^2\Delta t^2}{\Delta x^2} K\mathbf{p}^{(n)}}}^{\text{FDTD Laplacian}} + \Delta t^2 \mathbf{f}^{(n)}. \tag{46}$$

The Laplacians annotated with boxes above are plotted in Figure 4 for the second-order interface scheme applied to the second-order FDTD method and the first and second-order SEM. The two methods are running simultaneously in the entire domain (no coupling) with the residual subtracted corresponding to applying only Eq. (39) and (40), and leave out Eq. (41). This correspond to transforming from Neumann to Dirichlet boundary conditions with endpoints $p(-\Delta x, t) = p(l + \Delta x, t) = 0$ outside the domain. No interface errors are introduced in the FDTD simulation since the interface scheme and the simulation scheme have the same order. We note that much bigger errors are introduced for the higher-order SEM due to the shock introduced when enforcing Neumann boundaries at each time step, which can be observed both in the pressure plot and the plot of the Laplacians at $x = 0$. Therefore, a simple remedy is to add an SEM layer of first-order polynomials near the interface as illustrated in Figure 5, where the number of nodes in the layer should correspond to half the interface scheme order. In [10] [Section 5.6], filtering methods have been proposed to reduce the errors in the presence of shocks, and it should be investigated if this method can be applied to reduce the approximation errors for higher-order SEMs.

7 Source signals

When choosing an excitation signal, all input signals can be used as long as the signal is band-limited to fit the frequency range of the simulation. We are considering the normalized differentiated Gaussian

$$U_t(t) = \frac{t - t_0}{t_w} e^{-\frac{(t-t_0)^2}{t_w^2}}, \qquad U_x(x) = \frac{x - x_0}{x_w} e^{-\frac{(x-x_0)^2}{x_w^2}}, \tag{47}$$

where t_0 is the time delay, and x_0 is the center of the Gaussian. The constants t_w and x_w are determining the width of the Gaussian in the temporal and spatial dimensions[3] defined as

$$t_w = \frac{1}{(\pi \cdot \nu f_{\text{max}}/2)}, \qquad x_w = \frac{c}{(\pi \cdot \nu f_{\text{max}}/2)}, \tag{48}$$

[3] At each discrete time-step Δt, the duration must be less than the time it takes for the wave to travel to the next spatial point, i.e., $c\Delta t = \Delta x$, implying $x_w = ct_w$ for the spatial domain.

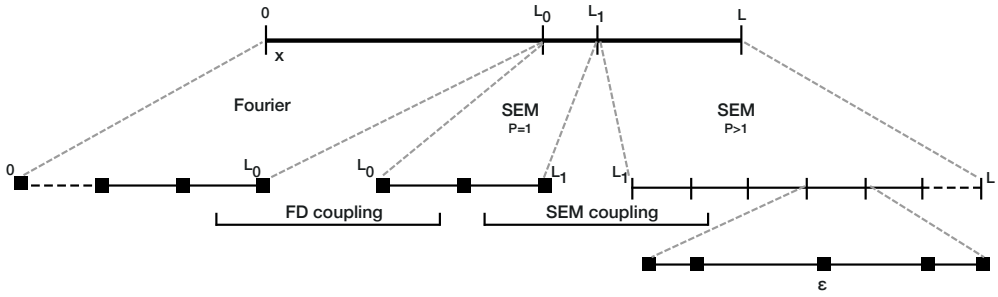


Figure 5: Domain partitioning coupling the Fourier method with the SEM including a tiny layer running the 1st order SEM.

where $t_0 \geq 4t_w$ is ensuring smooth signal initializations. The source taking both spatial and temporal dimensions into account is

$$F(x,t) = U_t(t)U_x(x). \tag{49}$$

When gradually injecting the time/space forcing function in Equation (49), the resulting wave propagation in the simulations has shown to not exactly span the expected frequencies determined from the width of the Gaussian function in Equation (48). Although the relation of t_w and x_w is properly formulated through the maximum frequency f_{max} and the speed of sound c, the resulting wavelength in the simulation is slightly larger compared to the spatial formulation in Equation (47). Therefore the constant $\nu \geq 1$ is introduced to adjust the frequencies. We have found that choosing $\nu_{\text{sim}} = 1.3$ for the injected source in Equation (49) is resulting in simulated waves with expected wavelengths.

8 Experiments

We will investigate the accuracy and efficiency of the domain decomposition method coupling the FM with the SEM. All experiments are done in a 1D domain with $f_{\text{max}} = 1000$ with the differentiated Gaussian pulse injected halfway into the FM partition 1. In the following, we will use the notation FM{ppw} and SEM{ppw}, i.e., FM4-SEM8 would be the FM-SEM coupling with spatial resolutions of four and eight ppw per partition, respectively.

To access the accuracy of the method, we will separately consider interface errors, and errors over time at a receiver position. The interface errors are investigate by measuring the reflected pressures in the vicinity left of interface for a left to right travelling wave at time $t_{\text{refl}} = 0.0135$ corresponding to the wave having made a single pass through the interface. The errors over time at the receiver position is compared against a reference solution for a total running time $t_{\text{max}} = 0.2$ s using different measures.

8.1 Interface errors

The domain length in this experiment is 10 m split into two partitions of 5 m each. We measure the reflected pressure errors in dB as

$$\epsilon_{\text{interface}} = 20 \log \left(\frac{\max(\mathbf{p}_{\text{refl}})}{p_{\text{max}}} \right) \text{ dB}, \tag{50}$$

where $\mathbf{p}_{\text{refl}} = \{p(t,x) \mid x \in [3.5, 5],\ t = 0.0135\}$ are the reflected pressure values in the vicinity of the interface after a single interface traversal and has been chosen ad hoc such that no other wavefronts are overlapping. The pressure values before traversing the interface are normalized by the maximum pressure p_{max} in the domain. The interface error is then the relation between the incident and the reflected wave where $\epsilon_{\text{interface}} = -\infty$ corresponds to zero pressures (no interface error) being reflected.

The interface errors for the FM-SEM and FM-FM couplings are summarized in Table 1a and 1b, respectively. The errors are measured for combinations of spatial resolutions in the partitions denoted by the number of ppw. Overall, we see that using three ppw for the FM shows large errors indicating that four ppw is the lowest resolution possible for reasonable accuracies. The FM can handle two ppw corresponding to the Nyquist criterium but is limited by the sixth-order FD scheme at the interface. FM4-FM4 gives -41 dB interface errors, which is among the lowest combinations for coupling FMs, whereas FM4-SEM8 gives the lowest interface error of -36 dB for coupling FM with SEM.

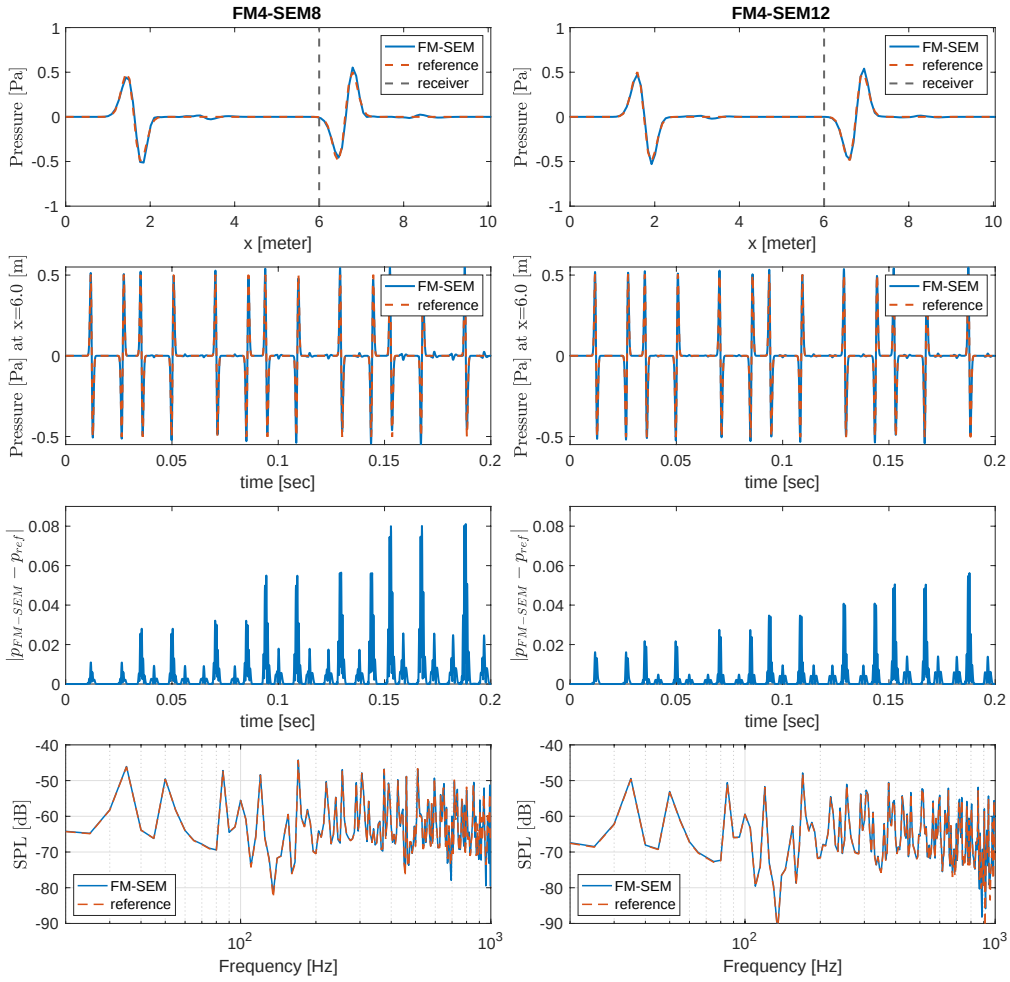


Figure 6: FM-SEM running until $t_{\max} = 0.2$ s. From the top: 1) Wave propagation in the full domain at $t = 0.2$ sec, 2) impulse response at receiver position $x = 6.0$ m, 3) L_1 error, 4) transfer function at $x_r = 6.0$. a) FM4-SEM8, b) FM4-SEM12.

Interface errors at $t = 0.0135$ sec

FM \ SEM	3	4	6	8	9	12
3	-25 dB	-26 dB†	-28 dB	-26 dB†	-26 dB	-25 dB
4	-	-34 dB	-35 dB†	-36 dB	-32 dB†	-32 dB

(a)

Interface errors at $t = 0.0135$ sec

FM1 \ FM2	3	4	6	8	9	12
3	-32 dB	-29 dB†	-24 dB	-27 dB†	-31 dB	-28 dB
4	-	-41 dB	-38 dB†	-30 dB	-40 dB†	-43 dB

(b)

Errors at $t = 0.2$ sec

FM↔SEM [ppw]	$\epsilon_{\text{rel\_rms}}$	ϵ_{rel}	ϵ_∞
$4 \leftrightarrow 4$	0.5%	-	0.2528
$4 \leftrightarrow 8$	1.3%	9.1%	0.0811
$4 \leftrightarrow 12$	2.4%	5.7%	0.0562

(c)

Table 1: Errors coupling the FM and the SEM using a sixth-order FDTD scheme for various spatial resolutions determined by the points per wavelengths (ppw). The differentiated Gaussian source is injected half-way into the left partition at $x_s = 2.5$ m (similar errors is achieved injecting the source in the right SEM partition). (a) FM-SEM interface errors, (b) FM-FM interface errors, c) Root-Mean-Square error ϵ_{RMS}, relative error ϵ_{rel}, and maximum error ϵ_∞ calculated by comparing with an exact Fourier reference solution. †: $\Delta t_1 = \Delta t_2$, since the spatial resolution is not a multiple of each other.

8.2 Overall accuracy

Another investigation of the accuracy is to compare the simulation with a reference solution. The error of the impulse responses (IRs) at receiver position $x = 6.0$ m is calculated in the relative Root-Mean-Squared $\epsilon_{\text{rel\_rms}}$, relative error ϵ_{rel} and the maximum error ϵ_∞ norms

$$\begin{aligned}
\epsilon_{\text{rel_rms}}(x) &= \frac{|p_{\text{sim,rms}}(x) - p_{\text{ref,rms}}(x)|}{p_{\text{ref,rms}}(x)}, \\
\epsilon_{\text{rel}}(x) &= \frac{1}{T}\sum_{n=0}^{T-1} \frac{|p_{\text{sim}}(t_n, x) - p_{\text{ref}}(t_n, x)|}{p_{\text{ref}}(t_n, x)}, \\
\epsilon_\infty(x) &= \max_{n=0,1,\ldots,T-1} |p_{\text{sim}}(t_n, x) - p_{\text{ref}}(t_n, x)|,
\end{aligned} \tag{51}$$

where

$$p_{\text{rms}}(x) = \sqrt{\frac{1}{T}\sum_{n=0}^{T-1} |p(t_n, x)|^2}, \tag{52}$$

(53)

is the Root-Mean-Squared (RMS). The simulated pressures $p_{\text{sim}}(t_n, x)$ and the reference pressures $p_{\text{ref}}(t_n, x)$ correspond to the IRs at the receiver position x for the discrete time steps t_n, and $T = \lceil t_{\text{max}}/\Delta t \rceil$ is the total number of time steps. The results are summarized in Table 1c. First, we notice that FM4-SEM4 gives the larges errors $\epsilon_\infty = 0.25$ and very big relative errors (not included in the table). In contrary, coupling with SEM8 or SEM12 drastically improves the accuracy with errors of $\epsilon_\infty < 0.08$ and $\epsilon_{\text{rel}} < 9\%$. These results are not all consistent with the interface errors, notice for example that the FM4-SEM8 coupling (-36 dB) compared to FM4-SEM12 (-32 dB) has 4 dB lower reflection errors, which indicates that the SEM is introducing numerical dispersion errors contributing to the overall error. Using other time integration schemes, such as the Runge-Kutta method, should decrease the SEM numerical errors.

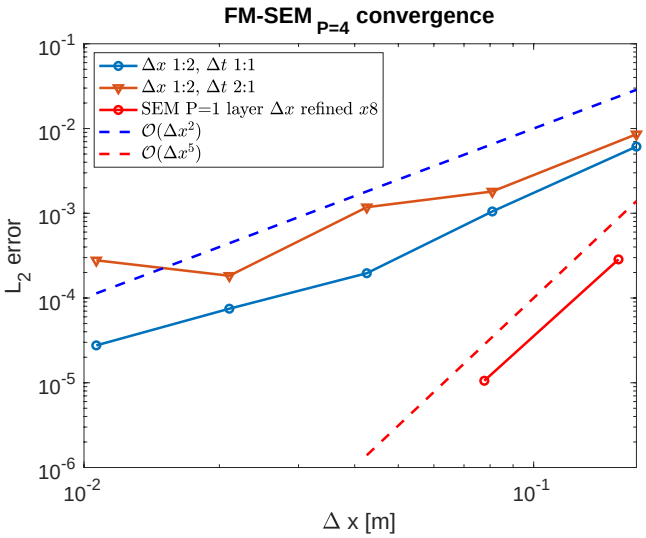


Figure 7: h-Convergence plots: FM{2,4,8,16,32}-SEM{4,8,16,32,64} are plotted pair-wise in the two top-most graphs, where 1) the temporal resolution is fixed for the blue graph, and 2) the temporal SEM resolution is twice the FM resolution for the blue graph. The lower red graph 3) shows convergence according to polynomial order $P = 4$ when oversampling 8 times the SEM $P = 1$ interface layer compared to the main SEM $P = 4$.

In Figure 6, the FM4-SEM8 and FM4-SEM12 results are plotted against the reference solution. The first row shows the pressure in the domain at $t_{\text{max}} = 0.2$ s, the second and forth row show the IR and the transfer function (TF) at receiver position $x = 6.0$ m and the third row shows the L_1 IR errors over time. We see an excellent match between the simulation and the reference, though with some small noticeable pressure perturbations in the wave propagation plot (top) due to interface and SEM dispertion errors.

8.3 Convergence

In Figure 7 the convergence is plotted pair-wise for FM{2,4,8,16,32}-SEM{4,8,16,32,64} with 1) fixed temporal resolution, 2) individual temporal resolution using Eq. (17), and 3) pair-wise for FM{16,32}-SEM{16-2,32-4} where the SEM $P = 1$ interface layer is oversampled eight times compared to the main SEM with fixed temporal resolution. When the $P = 1$ layer is finely oversampled, the spectral convergence is preserved, indicating that the interface errors are converging at the same rate.

8.4 Efficiency

The theoretical speedup in 3D for a large FM-SEM domain ratio is $2 \times x^3$, where x is the spatial resolution factor between the FM and the SEM running in the two partitions, resulting in a $16\times$ speedup for FM4-SEM8 and $54\times$ speedup for FM4-SEM12. The factor of 2 stems from the time resolution being twice as coarse for the FM. On top of that, the SEM time complexity for solving the system of equations consisting of sparse band matrices can be done in $\mathcal{O}(q^2 n)$ + $\mathcal{O}(qn)$ in time and $\mathcal{O}(n(2q + 1))$ in memory using direct solvers, where q is the bandwidth of the matrix and n is the degrees of freedom. The Fourier method is $\mathcal{O}(N \log(N))$ in time and n in memory when using the Fast Fourier Transform.

We will perform an empirical evaluation of the efficiency gained from the FM-SEM coupling compared to running the SEM in the entire domain for $t_{\text{max}} = 0.2$ s. The methods are implemented in Matlab, and the timings exclude the matrix assembly. The vast majority of the time in the SEM is spent solving the linear system of equations and is implemented using the Matlab backslash operator $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$ for solving the linear system $\mathbf{Ax} = \mathbf{b}$. For the FM, most of the work is spent in the Fourier transformation, where the build-in Matlab function 'FFT' is used. In all experiments, we will compare against the baseline forth-order SEM with Neumann boundaries.

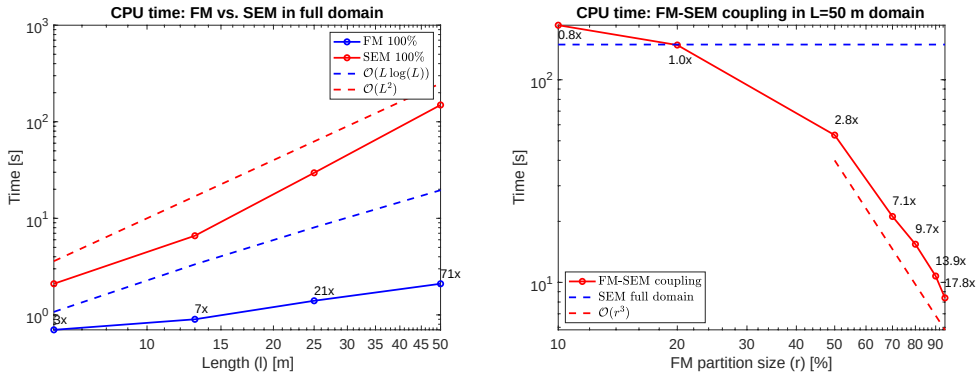
FM4-SEM8 CPU timings for $l = 50$ m

FM partition size (r)	FM solver	SEM solver	interface
50%	1.2%	84.9%	13.0%
80%	6.0%	53.6%	39.6%
95%	10.1%	18.5%	70.3%

Table 2

In the first experiment, we compare the CPU time separately for the FM and SEM running in the entire 1D domain of lengths $l = [6, 13, 25, 50]$ m, i.e. with no couplings. The result is shown in Figure 8a, and we see 3x to 71x speedups depending on the domain size. The CPU times scale with $\mathcal{O}(l^2)$ for the SEM and below the theoretical $\mathcal{O}(l \log(l))$ limit for the FM.

In the second experiment, we fix the domain size to 50 m and compare the CPU time for the FM-SEM coupling for different Fourier partition sizes of $r = [10, 20, 50, 70, 80, 90, 95]$ % relative to the entire domain. In Figure 8b the results are depicted, and we achieve speedups between 2x and 17x for Fourier partition sizes above 50% with scaling close to $\mathcal{O}(r^3)$. In Table 2, the timings for $l = 50$ m are shown separately for the FM and SEM solvers



(a) (b)

Figure 8: CPU timings. a) Comparison between the FM and the SEM running in the full domain for different domain sizes (no couplings), b) Comparison between FM-SEM and a baseline SEM with different relative FM partition sizes.

and the interface handling. We notice that the SEM workload for a 50/50 partition split is taking the 85% of total computation time. Increasing the partition size of FM drastically decreases the SEM workload and for $r = 95$ %, the workload of the interface handling starts dominating taking up 70% of the time. Most of the workload at the interface is because of the space and time spline interpolation, though there is significant overhead when calling the interpolation methods interpolating only a few points near the interface. In fact, interpolating all pressure values instead of only the values around the interface has a minor impact on the absolute performance. Therefore, we expect more time-efficient interpolations when extending to 2D and 3D, where much bigger pressure grids are to be interpolated near the interface.

Conclusion

We have implemented an SEM and a FM coupled at the interface using a (2,6) FDTD scheme handling independent spatial and temporal resolutions. Coupling the Fourier method using four points per wavelength with the SEM using eight points per wavelength in a 5 m + 5m domain results in -36 dB interface errors and 9.1% relative errors compared to a reference solution. Using 12 points per wavelength for the SEM gives slightly better relative errors of 5.7%. The efficiency of the coupled method is compared against an SEM running in the entire domain. For a fixed domain size of 50 m, the efficiency of the coupled method was compared for different relative FM partition sizes, with a 18 times performance gain when 95% of the domain is running the FM. A more significant performance gain is expected to be achieved when going to larger 2D and 3D domains due to more degrees of freedom to be handled by the SEM.

However, the workload at the interface will also grow compared to 1D, but we expect it to still be negligible compared to the saved computation time running the more expensive SEM solver.

Future work could consider the use of better time-stepping schemes, such as the Runge-Kutta methods, improving the accuracy of the SEM method. A limitation of the implemented method is the need for an additional SEM layer of first-order polynomials dominating the overall convergence rate. However, the layer is needed due to the shocks arising from the enforced (non-physical) Neumann boundary condition at every time step, causing large errors in the discrete Laplacian operator only for higher-order SEM. In [10] [Section 5.6], filtering methods have been proposed to reduce the errors in the presence of shocks, and it should be investigated if this method can be applied to minimize the errors.

References

A Derivation of the update scheme described in terms of modes

Reinterpreting the wave equation in a discrete setting for the spatial dimensions and taking the cosine transform of both sides of the equality sign yields

$$\begin{aligned}
\text{DCT}(F(\mathbf{x},t)) &= \text{DCT}\left(c^2\nabla^2 p(\mathbf{x},t) - \frac{\partial^2}{\partial t^2}p(\mathbf{x},t)\right) && (54)\\
&= \text{DCT}\left(c^2\nabla^2 p(\mathbf{x},t)\right) - \text{DCT}\left(\frac{\partial^2}{\partial t^2}p(\mathbf{x},t)\right) && (55)\\
&= -c^2 k_i^2 M_i - \frac{\partial^2}{\partial t^2}M_i, && (56)
\end{aligned}$$

Ignoring the forcing term, the above equation describes a set of independent simple harmonic oscillators, each vibrating with its own characteristic frequency $\omega_i = ck_i$. We have used the iDCT from Equation (11), the Linearity Theorem and the fact that the DCT of derivatives is given as $\text{DCT}(f''(x)) = -k^2 f(k)$, since double-differencing the inverse cosine transform, $\mathcal{F}^{-1}$, of $F(x)$ gives

$$\begin{aligned}
\frac{d^2}{dx^2}\sqrt{\frac{2}{\pi}}\int F(\omega)\cos(\omega x)d\omega &= \frac{d}{dx}\sqrt{\frac{2}{\pi}}\int F(\omega)\frac{d}{dx}\cos(\omega x)d\omega && (57)\\
&= -\sqrt{\frac{2}{\pi}}\int F(\omega)\omega\frac{d}{dx}\sin(\omega x)d\omega && (58)\\
&= -\omega^2\sqrt{\frac{2}{\pi}}\int F(\omega)\cos(\omega x)d\omega && (59)\\
&= \mathcal{F}^{-1}(-\omega^2 F(\omega)) \Rightarrow && (60)\\
\mathcal{F}(f''(x)) &= -\omega^2 F(\omega) && (61)
\end{aligned}$$

B Derivation of the update scheme for a time-dependent source terms

By "reverse engineering" the final update scheme in [22], we notice that the time derivate has been multiplied by the term $\frac{\omega_i^2\Delta t^2}{2(1-\cos(\omega_i\Delta t))}$ as

$$\frac{M_i^{(n+1)} - 2M_i^{(n)} + M_i^{(n-1)}}{\Delta t^2}\frac{\omega^2\Delta t^2}{2(1-\cos(\omega_i\Delta t))} + \omega_i^2 M_i^{(n)} - \tilde{F}_i^{(n)} = 0 \tag{62}$$

By taking a closer look at this term, we see that it evaluates to $1 + O(\Delta t^2)$, since by doing a Taylor expansion of $\cos(\omega_i\Delta t)$ around Δt, we get $\cos(\omega_i\Delta t) = 1 - \frac{1}{2}(\omega_i\Delta t) + O(\Delta t^4)$:

$$\begin{aligned}
\frac{\omega_i^2\Delta t^2}{2(1-\cos(\omega_i - \Delta t))} &= \frac{\omega_i^2\Delta t^2}{2(1-1+\frac{1}{2}(\omega_i\Delta t) + O(\Delta t^4))} && (63)\\
&= \frac{\omega_i^2\Delta t^2}{\omega_i^2\Delta t^2 + O(\Delta t^4)} && (64)\\
&= \frac{1}{1+O(\Delta t^2)} && (65)\\
&= 1 + O(\Delta t^2) \to 1, \quad \text{for } \Delta t \to 0 && (66)
\end{aligned}$$

where $\frac{1}{1+x} = 1 + x + O(x^2)$ has been used in the last step. Hence, multiplying the time derivative by these terms corresponds to multiplying by 1, and since the truncation error is no greater than the error introduced by the centered difference, this approximation is not making the solution worse. The term originates from the solution to the simple simple harmonic oscillator $\frac{\partial^2 M}{\partial t^2} = -\omega_i^2 M$ with the solution $M_i(t) = C\cos(\omega_i t)$. Finally, we isolate $M_i^{(n+1)}$ giving us the update equation

$$M_i^{(n+1)} = 2M_i^{(n)}\cos(\omega_i\Delta t) - M_i^{(n-1)} + \frac{2\tilde{F}_i^{(n)}}{\omega_i^2}(1-\cos(\omega_i\Delta t)) \tag{67}$$

Report B

A summary of the wave-based method (WBM) with convergence studies

Nikolas Borrel-Jensen

November 9, 2023

Contents

1 Introduction

The wave-based method (WBM) has been recognized to offer superior convergence compared to standard boundary element methods and finite-element methods. The literature lacks theoretical convergence proofs, and therefore an investigation examining its convergence compared to state-of-the-art methods, such as the spectral-element method, and the results are presented in Section 6.

This note aims to provide a concise overview of the WBM, drawing from relevant literature sources [**?**,**?**,**?**,**?**]. Additionally, the Green's function solution to the Helmholtz equation is presented in this text, serving as a means to validate the solutions.

2 Problem formulation

The WBM is designed for solving steady-state problem described by the Helmholtz equation

$$\nabla^2 p(\mathbf{r}) + k^2 p(\mathbf{r}) = f(\mathbf{r}), \qquad \mathbf{r} \in \Omega \tag{1}$$

where ∇^2 is the Laplacian, $p(\mathbf{r})$ is the pressure field variable of the Helmholtz equation, $k = \omega/c$ is the wave number, c the speed of sound, $f(\mathbf{r})$ is the forcing function and Ω is the problem domain. Consider the bounded domain $\Gamma = \partial\Omega$ depicted in Figure (1) divided into three non-overlapping parts $\Gamma = \Gamma_v \cup \Gamma_p \cup \Gamma_Z$, where the following boundary conditions apply

$$\mathbf{r} \in \Gamma_v : \mathcal{L}_v(p(\mathbf{r})) = \bar{v}_n(\mathbf{r}) \tag{2a}$$

$$\mathbf{r} \in \Gamma_p : p(\mathbf{r}) = \bar{p}_n(\mathbf{r}) \tag{2b}$$

$$\mathbf{r} \in \Gamma_Z : \mathcal{L}_v(p(\mathbf{r})) = \frac{p(\mathbf{r})}{\bar{Z}(\mathbf{r})} \tag{2c}$$

where $\bar{v}_n(\mathbf{r})$, $\bar{p}_n(\mathbf{r})$ and $\bar{Z}(\mathbf{r})$ are the imposed normal velocity (Neumann), pressure (Dirichlet) and normal impedance at the boundaries, respectively and the normal velocity operator $\mathcal{L}_v$ is defined as

$$\mathcal{L}_v(\bullet) = \frac{j}{\rho\omega}\frac{\partial \bullet}{\partial n} = \frac{j}{\rho\omega}(\mathbf{n})^T \nabla(\bullet) \tag{3}$$

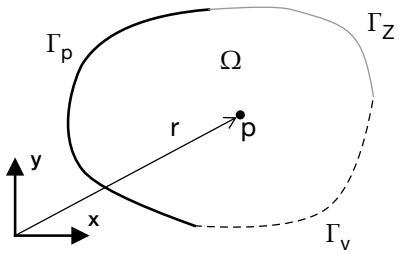


Figure 1: Domain consisting of three boundary types (pressure, velocity, impedance)

3 Greens function

We will compare our simulations to the Green's function to the Helmholtz equation using a point source as forcing function modelled by the dirac delta function $\delta(\cdot)$. The Helmholtz equation with the point source in $\mathbf{r}_0$ is then

$$\nabla^2 p(\mathbf{r}) + k^2 p(\mathbf{r}) = -\delta(\mathbf{r} - \mathbf{r}_0) \tag{4}$$

for which the analytical solution in an enclosure is given by the Greens function [**?**]

$$G(\mathbf{r}, \mathbf{r}_0) = -\frac{1}{V}\sum_m \frac{\psi_m(\mathbf{r})\psi_m(\mathbf{r}_0)}{k^2 - k_m^2} \tag{5}$$

For Neumann boundary conditions, ψ_N is given as

$$\psi_N(x,y) = \sqrt{\epsilon_{n_x}\epsilon_{n_y}} \cos\left(\frac{n_x \pi x}{l_x}\right) \cos\left(\frac{n_y \pi y}{l_y}\right) \tag{6}$$

and for Dirichlet boundary conditions as

$$\psi_N(x,y) = \sqrt{\epsilon_{n_x}\epsilon_{n_y}} \sin\left(\frac{n_x \pi x}{l_x}\right) \sin\left(\frac{n_y \pi y}{l_y}\right) \tag{7}$$

where l_x and l_y are the domain size in x and y, respectively, n_x, n_y are wave function index $0, 1, 2, ..$, the term $\sqrt{\epsilon_{n_x}\epsilon_{n_y}}$ is a normalization constant with $\epsilon_m = 1$ for $m = 0$ and $\epsilon_m = 2$ for $m > 0$ and $k_m^2 = \sqrt{\left(\frac{n_x \pi}{l_x}\right)^2 + \left(\frac{n_y \pi}{l_y}\right)^2}$ is the wave number for the modes/natural frequencies in the enclosure. N is representing the two integers n_x, n_y as $\sum_m^N = \sum_{n_x=0}^{\infty} \sum_{n_y=0}^{\infty}$.

4 The spectral-element method

The SEM is derived for approximating the solution to the Helmholtz equation with the aim of coupling with the WBM. The derivation follows the receipe from the lecture notes in [**?**].

4.1 Helmholtz derivation

We represent the problem formulation of (1) in weak form by applying integration by parts

$$-\int_\Omega \frac{\partial p}{\partial x}\frac{\partial v}{\partial x}\mathrm{d}x - \int_\Omega \frac{\partial p}{\partial y}\frac{\partial v}{\partial y}\mathrm{d}y + \int_\Omega\int_\Omega k^2 pv\mathrm{d}x\mathrm{d}y = \int\int_\Omega fv\mathrm{d}x\mathrm{d}y \tag{8}$$

where $v(x,y)$ is a test function chosen such that it vanishes at the endpoints. A truncated series expansion for the unknown variable p is introduced

$$p(x,y) \approx \hat{p}(x,y) = \sum_{i=1}^{K} \hat{p}_i N_i(x,y) \tag{9}$$

where $N_i(x,y)$ is the set of global finite element basis functions chosen as Lagrange polynomials. Insert Eq. (9) for p into Eq. (8), set the test function $v(x,y)$ equal to each of the basis functions for $N_j(x,y)$ and collect the terms into $\mathbf{A}\hat{p} = \mathbf{b}$. The matrix $\mathbf{A}$ can then be written

$$\mathbf{A}^{(n)} = k^2\mathcal{M}^{(n)} - \mathcal{S}^{(n)} \tag{10}$$

where $\mathcal{M}_{i,j}^{(n)}$ and $\mathcal{S}_{i,j}^{(n)}$ are the mass and stiffness matrices, respectively, for the (i,j) indexes of the n'th element

$$\begin{aligned}
\mathcal{M}_{ij}^{(n)} &= \int\int_{e_n} N_i^{(n)}(x,y) N_j^{(n)}(x,y)\mathrm{d}x\mathrm{d}y \\
\mathcal{S}_{x,ij}^{(n)} &= \int_{e_n} \frac{\partial N_i^{(n)}(x,y)}{\partial x}\frac{\partial N_j^{(n)}(x,y)}{\partial x}\mathrm{d}x \\
\mathcal{S}_{y,ij}^{(n)} &= \int_{e_n} \frac{\partial N_i^{(n)}(x,y)}{\partial y}\frac{\partial N_j^{(n)}(x,y)}{\partial y}\mathrm{d}y
\end{aligned} \tag{11}$$

The integrations in (11) can be calculated exact without resorting to e.g. Gaussian quadratures. It is convenient to introduce a reference triangle element [**?**]

$$\mathcal{I} = \{(r,s) | (r,s) \geq -1 \leq r, s; r + s \leq 0\}$$

The nodal Lagrange basis functions $N_n(r,s)$ can be determined on the reference element $\mathcal{I}$ using the Vandermonde matrix $\mathcal{V}$

$$N_n(r,s) \quad = \quad \sum_{n=1}^{P+1} \left(\mathcal{V}^T\right)^{-1}_{i,n} \psi_n(r,s) \tag{12}$$

where $\psi_n(r,s)$ are modal Legendre polynomials and the nodal distribution of the collocation points r,s of the reference element is of the Legrende-Gauss-Lobatto (LGL) kind. Inserting the above into the expression for the mass matrix in (11), but on the reference element yields

$$\mathcal{M} = (\mathcal{V}\mathcal{V}^T)^{-1} \tag{13}$$

where $\mathcal{V}$ is the Vandermonde matrix. The mapping from the local reference element $\mathcal{I}$ to the global element e_n on (x,y) in 11 is

$$\begin{aligned} \mathcal{M}^{(n)}_{ij} = \int\int_{e_n} N^{(n)}_i(x,y) N^{(n)}_j(x,y) \mathrm{d}x\mathrm{d}y = \\ \mathcal{J}^n \int\int_{\mathcal{I}} N^{(n)}_i(r,s) N^{(n)}_j(r,s) dr ds \end{aligned} \tag{14}$$

where $\mathcal{J}^n$ is the Jacobian mapping from local to the global element n as $(x,y) \rightarrow (r,s)$. Regarding the stiffness matrix, we use

$$\frac{\partial}{\partial r} N_i(r,s) = \sum_{n=1}^{P+1} \frac{\partial}{\partial r} N_i(r_n, s_n) N_n(r,s) \tag{15}$$

and inserting the above term into the expression for the stiffness matrix, but (again) on the reference element yields

$$\mathcal{S}_r = \mathcal{D}_r^T \mathcal{M} \mathcal{D}_r, \quad \mathcal{S}_s = \mathcal{D}_s^T \mathcal{M} \mathcal{D}_s \tag{16}$$

where $\mathcal{D}_r$ is the differentiation matrix

$$\mathcal{D}_r \quad = \quad \mathcal{V}_r \mathcal{V}^{-1} \tag{17}$$

The mapping from the reference element to the global element n for the stiffness matrix is again done using the Jacobian $\mathcal{J}^n$

$$\mathcal{S}^{(n)} = \mathcal{J}^n \mathcal{D}_x^T \mathcal{M} \mathcal{D}_x + \mathcal{J}^n \mathcal{D}_y^T \mathcal{M} \mathcal{D}_y \tag{18}$$

where

$$\mathcal{D}_x = r_x \mathcal{D}_r + s_x \mathcal{D}_s, \quad \mathcal{D}_y = r_y \mathcal{D}_r + s_y \mathcal{D}_s \tag{19}$$

and r_x, r_y, s_x, s_y are geometrical factors. To summarize, the mass and stiffness matrices for the n'th element can be calculated as

$$\mathcal{M}^{(n)} = \mathcal{J}^{(n)} (\mathcal{V}\mathcal{V}^T)^{-1}, \quad \mathcal{S}^{(n)} = \mathcal{J}^{(n)} \mathcal{D}_x^T \mathcal{M} \mathcal{D}_x \tag{20}$$

Similarly, for the elements of $\mathbf{b}$ resulting from the forcing function, we get the local constribution for element n as

$$b^{(n)} = \int\int_{e^n} f v \mathrm{d}x\mathrm{d}y = \mathcal{M}^{(n)} f^{(n)} \tag{21}$$

4.2 Point source implementation

In applications concerned with far-field properties of the pressure field, the point source can be used as source function [**?**]

$$f(x,y) = \delta(x - x_0, y - y_0) \tag{22}$$

where $(x0, y0)$ is the coordinate of the point source and $\delta(\cdot)$ is the delta function. The most straightforward implementation is to avoid the integration derived in 21 for the right hand-side source term in the weak formulation 8 and instead exploit the properties of the integration of Dirac delta functions as

$$\int\int_{\Omega} f(x,y)N_i(x,y)\mathrm{d}x\mathrm{d}y = \int\int_{\Omega} \delta(x-x_0, y-y_0)N_i(x,y)\mathrm{d}x\mathrm{d}y = N_i(x_0,y_0) \tag{23}$$

which on the reference element corresponds to the Lagrange polynomial calculated using Eq. (12) repeated here

$$N_n(r,s) = \sum_{n=1}^{P+1} \left(\mathcal{V}^T\right)^{-1}_{i,n} \psi_n(r,s)$$

Remember that the Vandermonde matrix is defined as

$$\mathcal{V}_{ij} = \psi_j(\mathbf{r}_i), \quad 1,\ldots,P+1 \tag{24}$$

where $\psi_j(\mathbf{r}_i)$ are the modal Legendre polynomials evaluated on the Legrende-Gauss-Lobatto nodes. Therefore

$$N_i(\mathbf{r}_j) = \delta_{ij} \tag{25}$$

due to the properties of the Lagrange polynomials. Exploiting the result in 25 makes the point source implementation simple. The right-hand side corresponding to the expression in Eq. (21) but for point sources is then

$$\mathcal{N}^{(n)} = \mathcal{J}^{(n)}\mathcal{N} \tag{26}$$

It follows that in case the point source coincides with a grid point (not GLL points inside an element), only one coefficient will be non-zero. In the implementation it is ensured that a point source will always coincide with a grid point.

4.3 Pressure level

The point source is modelled as the Dirac Delta δ equal to zeros everywhere except at $x = 0$ and the integral over the entire area is equal to 1 $\int \delta(x)\mathrm{d}x = 1$. In practice, we set the amplitude at $x = 0$ to 1, but this means that the energy injected will depend on the grid resolution: remember that only one grid point is non-zero and depending on the element size, the elements sharing this grid point might not integrate to 1.

Pressure adjustment

To ensure a normalized injected energy, the amplitude of the point source should be adjusted such that the elements sharing the grid point of the source integrates to 1. Replacing the Jacobian factor $\mathcal{J}$ in Eq. (26) has worked for specific grid sizes, but no general solution has been found.

Ad hoc normalization

The solution for normalizing the pressure to the analytical Greens function has been done ad hoc by finding the highest pressures in the simulated and analytical solution and calculating the ratio R. The ratio is multiplied to all pressure values in the simulation. Calculating the mean of the differences has been tried out, but with less good results.

4.4 Mesh

Mesh creation with more resolution around the point source has been implemented using the meshing tool 'DistMesh'. An example can be seen in Figure 2.

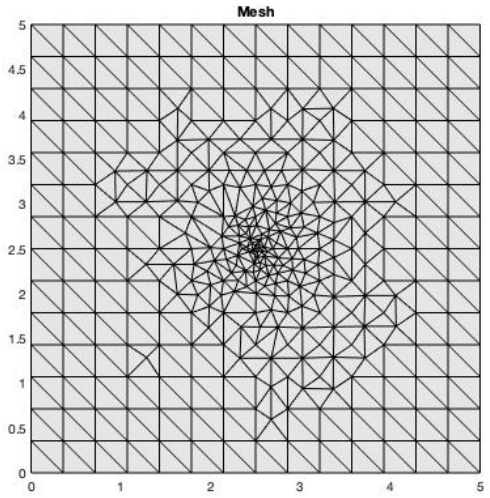


Figure 2: Refined mesh at point source location.

4.5 Grid resolution

The spatial resolution is dependent on the maximum simulation frequency. We have the following relations between wavelength $\lambda_{\min}$, domain size l, number of modes M and max frequency $f_{\max}$

$$\lambda_{\min} = \frac{c}{f_{\max}}, \tag{27a}$$

$$\Delta x = \frac{\lambda_{\min}}{N} \times P, \tag{27b}$$

where N is the number of points per wavelength and P is the polynomial order. The numerical simulation needs more points per wavelength that what the Nyquist Theorem states, typically between 5 and 15 points per wavelength.

5 WBM modelling

The procedure for solving the Helmholtz equation consists of four steps

1. Partition the problem into convex subdomains
2. Selection of a suitable set of wave functions for each subdomain
3. Construct and solve the system of matrices yielding the wave function contribution factors
4. Postprocessing of the dynamic variables

5.1 Domain partitioning into convex subdomains

Convexity is a sufficient condition for the WB approximation to converge towards the exact solution and hence non-convex domain Ω should be partitioned into convex non-overlapping subdomains as depicted in Figure 3 where $\Omega^{(\alpha)}, \alpha = 1, \ldots, N_\Omega$ and let $\Gamma_I^{(\alpha,\beta)}$ being the interface between two subdomains $\Omega^{(\alpha)}$ and $\Omega^{(\alpha)}$ such that

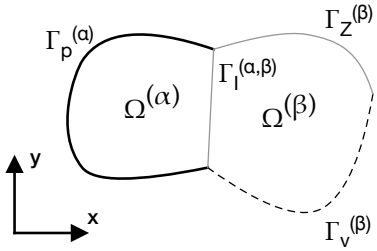


Figure 3: Domain partitioning

the a continuity condition is imposed on the interface. Pluymers [**?**] has shown that impedance coupling has better stability over directly enforcing pressure/velocity coupling

$$\mathbf{r} \in \Gamma_I : \left(\mathcal{L}_v^{(\alpha)}(p^{(\alpha)}(\mathbf{r})) - \frac{p^{(\alpha)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} \right) = - \left(\mathcal{L}_v^{(\beta)}(p^{(\beta)}(\mathbf{r})) + \frac{p^{(\beta)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} \right) \tag{28}$$

5.2 Wave functions

The steady-state fields $p(\mathbf{r})$ in each subdomain are approximated by a solution expansion $\hat{p}_w(\mathbf{r})$ in terms of the n_w number of wave functions ϕ_w for an α partition

$$p^{(\alpha)}(\mathbf{r}) \simeq \hat{p}^{(\alpha)}(\mathbf{r}) = \sum_{w=1}^{n_w^{(\alpha)}} p_w^{(\alpha)} \phi_w(\mathbf{r})^{(\alpha)} + \hat{p}_q^{(\alpha)}(\mathbf{r}) = \mathbf{\Phi}^{(\alpha)}(\mathbf{r}) \mathbf{p}_w^{(\alpha)} + \hat{p}_q^{(\alpha)}(\mathbf{r}), \tag{29}$$

where $\hat{p}_q^{(\alpha)}(\mathbf{r})$ represents a particular solution resulting from the forcing term in the Helmholtz equation, and $\mathbf{r}$ are the spatial coordinates. Each wave function $\phi(\mathbf{r})$ exactly satisfies the homogeneous part of the Helmholtz equation.

There are two choices of wave functions for 2D bounded domains, the r- and s-set

$$\sum_{w=1}^{n_w^{(\alpha)}} p_w^{(\alpha)} \phi_w^{(\alpha)}(\mathbf{r}) = \sum_{w_r=1}^{n_{w_r}^{(\alpha)}} p_{w_r}^{(\alpha)} \phi_{w_r}^{(\alpha)}(\mathbf{r}) + \sum_{w_s=1}^{n_{w_s}^{(\alpha)}} p_{w_s}^{(\alpha)} \phi_{w_s}^{(\alpha)}(\mathbf{r}) \tag{30}$$

The wave functions are defined as

$$\phi_w^{(\alpha)}(\mathbf{r}) = \begin{cases} \phi_{w_r}^{(\alpha)}(x,y) = \cos(k_{xw_r}^{(\alpha)} x) e^{-jk_{yw_r}^{(\alpha)} y}, \\ \phi_{w_s}^{(\alpha)}(x,y) = e^{-jk_{xw_s}^{(\alpha)} x} \cos(k_{yw_s}^{(\alpha)} y) \end{cases} \tag{31}$$

The only requirement for Eq. (31) to be an exact solution to the Helmholtz equation is, that the wave numbers satisfy

$$(k_{xw_r}^{(\alpha)})^2 + (k_{yw_r}^{(\alpha)})^2 = (k_{xw_s}^{(\alpha)})^2 + (k_{yw_s}^{(\alpha)})^2 = k^2$$

There are infinitely many solutions for the above relation, but the following have been proposed

$$\begin{aligned} (k_{xw_r}^{(\alpha)}, k_{yw_r}^{(\alpha)}) &= \left(\frac{w_1^{(\alpha)}}{L_x^{(\alpha)}}, \pm\sqrt{k^2 - \left(k_{xw_r}^{(\alpha)}\right)^2} \right) \\ (k_{xw_s}^{(\alpha)}, k_{yw_s}^{(\alpha)}) &= \left(\pm\sqrt{k^2 - \left(k_{yw_s}^{(\alpha)}\right)^2}, \frac{w_2^{(\alpha)}}{L_y^{(\alpha)}}, \right) \end{aligned}$$

$L_x^{(\alpha)}$ and $L_y^{(\alpha)}$ are the dimensions of the smallest box surrounding the subdomain as depicted in Fig. 4, $k = \omega/c$, ω being the angular frequency, and $w_1 = w_2 = 0, 1, \ldots$. This choice of wave functions leads to standing and evanescent waves.

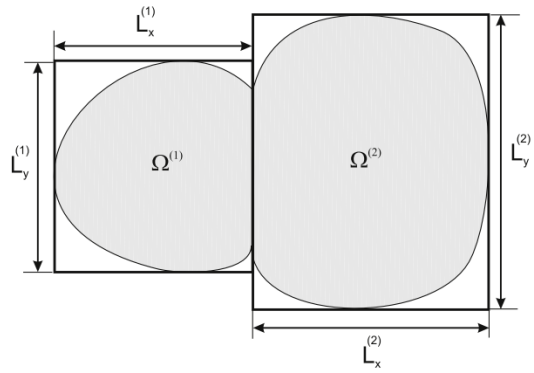


Figure 4: Domain enclosed by the smallest box. Figure from [**?**].

5.2.1 Truncation

In order to apply the WB formulation in a numerical scheme, the wave numbers w_1 and w_2 should be bounded and hence truncating the wave functions. A linear, frequency-dependent truncation rule is suggested, where wave functions that have wave number components smaller or equal a factor N times the physical wave number k are added to the wave function basis set

$$\begin{aligned} \frac{w_{1_{max}}^{(\alpha)}}{L_x^{(\alpha)}} \simeq Nk \quad &\Rightarrow \quad w_{1_{max}}^{(\alpha)} = \lceil L_x^{(\alpha)} Nk \rceil \\ \frac{w_{2_{max}}^{(\alpha)}}{L_y^{(\alpha)}} \simeq Nk \quad &\Rightarrow \quad w_{2_{max}}^{(\alpha)} = \lceil L_y^{(\alpha)} Nk \rceil \end{aligned}$$

where $w_{1_{max}}, w_{2_{max}} \in \mathbb{N}$ and hence

$$w_1 \in [0, 1, \ldots, w_{1_{max}}] \quad \text{and} \quad w_2 \in [0, 1, \ldots, w_{2_{max}}]$$

The total number of wave functions in Eq. (30) is $n_w^{(\alpha)} = n_{w_r}^{(\alpha)} + n_{w_s}^{(\alpha)}$ in subdomain $\Omega^{(\alpha)}$ with

$$\begin{aligned} n_{w_r}^{(\alpha)} &= 2(w_{1_{max}} + 1) \\ n_{w_s}^{(\alpha)} &= 2(w_{2_{max}} + 1) \end{aligned}$$

5.2.2 Scaling

Some of the wave functions are oscillatory in all (2) dimension and are restricted in the interval $[-1, 1]$, whereas other wave functions are evanescent in one of the dimensions with amplitudes larger than 1. From an implementation point of view, scaling factors f_{yw_r} and f_{xw_s} are introduced to restrict all wave functions amplitudes below 1.

$$\begin{cases} \phi_{w_r}^{(\alpha)}(x, y) = \cos(k_{xw_r}^{(\alpha)} x) e^{-jk_{yw_r}^{(\alpha)}(y - f_{yw_r} L_y)}, \\ \phi_{w_s}^{(\alpha)}(x, y) = e^{-jk_{xw_s}^{(\alpha)}(x - f_{yw_s} L_x)} \cos(k_{yw_s}^{(\alpha)} y) \end{cases} \tag{32}$$

5.2.3 Particular solution

The term $\hat{p}_q$ in Eq. (29) is resulting from the source term q in the inhomogeneous Helmholtz equation (1) and can be represented as the free-field solution of a point source

$$\hat{p}_q(x, y) = \frac{\rho \omega Q}{4} H_0^{(2)}(kr_q) \tag{33}$$

where $Q = \int_V q \mathrm{dV}$ is the source strength, $H_0^{(2)}$ is the zero-order Hankel function of second kind and $r_q = \sqrt{(x-x_q)^2 + (y-y_q)^2}$ the distance to the source point.

5.3 Formulation of residuals

The solution expansion in Eq. (29) is always an exact solution to the Helmholtz equation (1) irrespective of the unknown wave function contribution factors p_w. To determine the values of these contributions, the boundary conditions (2a-2c) must be taken into account, introducing approximations to the solutions. Hence, there are no residuals involved for the partial differential equation (for the inner domain), only residuals $R(\bullet)$ exists for the boundary and interface modelling

$$r \in \Gamma_v : R_v(\mathbf{r}) = \mathcal{L}(\hat{p}(\mathbf{r})) - \bar{v}_n(\mathbf{r}), \tag{34a}$$

$$r \in \Gamma_p : R_p(\mathbf{r}) = \hat{p}(\mathbf{r}) - \bar{p}(\mathbf{r}), \tag{34b}$$

$$r \in \Gamma_Z : R_Z(\mathbf{r}) = \mathcal{L}(\hat{p}(\mathbf{r})) - \frac{\hat{p}(\mathbf{r})}{\bar{Z}_n(\mathbf{r})}, \tag{34c}$$

where again $\bar{v}_n(\mathbf{r})$, $\bar{p}(\mathbf{r})$ and $\bar{Z}_n(\mathbf{r})$ are the imposed normal velocity (Neumann), pressure (Dirichlet) and normal impedance at the boundaries, respectively.

5.4 Weighted residual formulation

For each subdomain, the boundary residuals (34a-34c) are orthogonalised with respect to a weighting function $\tilde{p}$ or its derivative as

$$\int_{\Gamma_v^{(\alpha)}} \tilde{p}^{(\alpha)}(\mathbf{r}) R_v^{(\alpha)}(\mathbf{r}) \mathrm{d}\Gamma + \int_{\Gamma_p^{(\alpha)}} \tilde{p}^{(\alpha)}(\mathbf{r}) R_p^{(\alpha)}(\mathbf{r}) \mathrm{d}\Gamma + \int_{\Gamma_Z^{(\alpha)}} -\mathcal{L}(\tilde{p}^{(\alpha)}(\mathbf{r})) R_Z^{(\alpha)}(\mathbf{r}) \mathrm{d}\Gamma = 0 \tag{35}$$

The weighting function $\tilde{p}^{(\alpha)}(\mathbf{r})$ is expanded in terms of the same set of acoustic wave functions used in the pressure expansion (29)

$$\tilde{p}^{(\alpha)}(\mathbf{r}) = \sum_{a=1}^{n_w^{(\alpha)}} \tilde{p}_a^{(\alpha)} \phi_a^{(\alpha)}(\mathbf{r}) = \mathbf{\Phi}^{(\alpha)}(\mathbf{r}) \tilde{\mathbf{p}}_w^{(\alpha)} \tag{36}$$

5.5 System of linear equations

Substitution of the pressure expansion from Eq. (29) and the weighting function (36) for subdomain $\Omega^{(\alpha)}$ into the weighted residual formulation (35) result in a set of n_w linear equations for each of the unknown wave functions

$$\mathbf{A}\mathbf{p}_w = \mathbf{f} \tag{37}$$

given as

$\mathbf{A}^{(\alpha)} = \mathbf{A}_p^{(\alpha)} + \mathbf{A}_v^{(\alpha)} + \mathbf{A}_Z^{(\alpha)}$, $(n_w^{(\alpha)} \times n_w^{(\alpha)})$ with

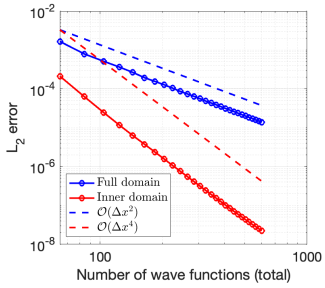
$$\mathbf{A}_v^{(\alpha)} = \int_{\Gamma_v^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{\Phi}^{(\alpha)T} \mathbf{n}^{(\alpha)T} \mathbf{B}^{(\alpha)} \mathrm{d}\Gamma \tag{38a}$$

$$\mathbf{A}_Z^{(\alpha)} = \int_{\Gamma_Z^{(\alpha)}} \left(\frac{j}{\rho\omega} \mathbf{\Phi}^{(\alpha)T} \mathbf{n}^{(\alpha)T} \mathbf{B}^{(\alpha)} - \frac{1}{\bar{Z}_n} \mathbf{\Phi}^{(\alpha)T} \mathbf{\Phi}^{(\alpha)} \right) \mathrm{d}\Gamma \tag{38b}$$

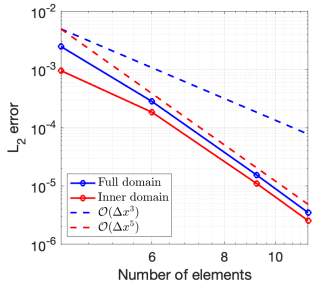
$$\mathbf{A}_p^{(\alpha)} = -\int_{\Gamma_p^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \mathbf{\Phi}^{(\alpha)} \mathrm{d}\Gamma \tag{38c}$$

$\mathbf{B}$ $(2 \times n_w^{(\alpha)})$, collecting the gradient components of the acoustic wave function $\mathbf{\Phi}^{(\alpha)}$

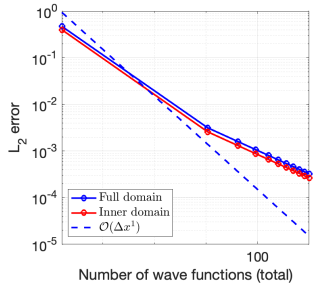
$$\mathbf{B} = \nabla \mathbf{\Phi}^{(\alpha)} \tag{39}$$



(a) WBM convergence as a function of wave functions $Nw = Nw_r + Nw_s = 8, 28, 48, \ldots, 608$ compared against SEM.



(b) SEM convergence as a function of elements compared against SEM.



(c) Green's function convergence as a function of total number of wave functions $Nw_x = Nw_y = 4, 104, \ldots, 1004$ per dimension compared against SEM.

Figure 5: L_2 convergence rate for ROIs using WBM, SEM and Green's.

$$\mathbf{f}^{(\alpha)} = \mathbf{f}_v^{(\alpha)} + \mathbf{f}_p^{(\alpha)} + \mathbf{f}_Z^{(\alpha)},\ (n_w^{(\alpha)} \times 1)$$

$$\mathbf{f}_v^{(\alpha)} = \int_{\Gamma_v^{(\alpha)}} \mathbf{\Phi}^{(\alpha)T} \left(\bar{v}_n^{(\alpha)} - \frac{j}{\rho\omega} \mathbf{n}^{(\alpha)T} \nabla \hat{p}_q^{(\alpha)} \right) \mathrm{d}\Gamma \tag{40a}$$

$$\mathbf{f}_Z^{(\alpha)} = \int_{\Gamma_Z^{(\alpha)}} \mathbf{\Phi}^{(\alpha)T} \left(\frac{\hat{p}_q^{(\alpha)}}{\bar{Z}_n} - \frac{j}{\rho\omega} \mathbf{n}^{(\alpha)T} \nabla \hat{p}_q^{(\alpha)} \right) \mathrm{d}\Gamma \tag{40b}$$

$$\mathbf{f}_p^{(\alpha)} = \int_{\Gamma_p^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \left(\hat{p}_q^{(\alpha)} - \bar{p}^{(\alpha)} \right) \mathrm{d}\Gamma \tag{40c}$$

See more details in [**?**, **?**].

5.6 Solution and postprocessing

After solving (37) for $\mathbf{p}_w$, the values are inserted back into the expansion in (29) yielding an approximation $\hat{p}$ of the acoustic pressure field. Derived acoustic fields can easily be obtained as well (see [**?**]).

6 Results

We compare the convergence rate solving the Helmholtz equation for perfectly reflecting boundaries with energy injected as a point source for 1) the Green's function, 2) the WBM, and 3) the Spectral Element Method (SEM). The convergence rate is examined in the full domain including boundaries and for the inner domain neglecting points near the boundary and the results can be seen in Figure 5. For all experiments we exclude points within a radius of $r = 0.4$ meters from the singularity in the source point. The reference solution is constructed using the SEM on a fine grid. We are calculating the steady-state solution at $f_{max} = 300$ Hz with point source location $s_{xy} = (1.2, 1.2)$ m and domain size $l_x = l_y = 2$m. We have investigated the convergence rate for the WBM, the SEM, and Green's function in RIOs to get insights about challenging parts. SEM shows convergence in accordance with theory for all ROIs. WBM has a 4th order convergence (L_2 error norm) when disregarding boundaries, corresponding to SEM using 3rd order polynomials. Green's function has a slow convergence of 1st order when disregarding the boundaries and axial directions, where an even slower convergence is observed.

Modeling pressure fields exited by point sources might be more challenging due to the abrupt injection, leading to highly oscillating waves. This requires more wave functions to capture the physics. A similar issue with degraded convergence has been observed when piston radiation from a boundary happens, resulting in discontinuous boundary conditions [**?**].

SEM does an excellent job, showing convergence rates according to theory. Though, calculating the first reflection using the (Green's) analytical solution (similar to BEM/WBM; sometimes known as 'background pressure field' in the literature) and injecting the resulting pressure as boundary conditions could increase the quality.

7 Multi-domain WBM

Two different methods can be used when coupling two WBM domains at the interface Γ_I: i) directly enforce pressure continuity on one subdomain and normal continuity on the other subdomain formulated or ii) apply impedance continuity conditions. The formulation w.r.t. pressure/normal continuity i) is given as

$$\text{normal vel. cont. } \mathbf{r} \in \Gamma_I^{(\alpha,\beta)} : \mathcal{L}_v^{(\alpha)}(\hat{p}^{(\alpha)}(\mathbf{r})) = -\mathcal{L}_v^{(\beta)}(\hat{p}^{(\beta)}(\mathbf{r})) \tag{41a}$$

$$\text{pressure cont. } \mathbf{r} \in \Gamma_I^{(\alpha,\beta)} : \hat{p}^{(\alpha)}(\mathbf{r}) = \hat{p}^{(\beta)}(\mathbf{r}) \tag{41b}$$

The impedance continuity condition ii) is given as

$$r \in \Gamma_I^{(\alpha,\beta)} : \mathcal{L}_v^{(\alpha)}(\hat{p}^{(\alpha)}(\mathbf{r})) - \frac{\hat{p}^{(\alpha)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} = -\mathcal{L}_v^{(\beta)}(\hat{p}^{(\beta)}(\mathbf{r})) + \frac{\hat{p}^{(\beta)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} \tag{42}$$

7.1 Formulation of residuals

The interface residuals error functions are given as

$$r \in \Gamma_{I,v} : R_{I,v}^{(\alpha,\beta)}(\mathbf{r}) = \mathcal{L}_v^{(\alpha)}(\hat{p}^{(\alpha)}(\mathbf{r})) + \mathcal{L}_v^{(\beta)}(\hat{p}^{(\beta)}(\mathbf{r})) \tag{43a}$$

$$r \in \Gamma_{I,p} : R_{I,p}^{(\alpha,\beta)}(\mathbf{r}) = \hat{p}^{(\alpha)}(\mathbf{r}) - \hat{p}^{(\beta)}(\mathbf{r}) \tag{43b}$$

$$\begin{aligned} r \in \Gamma_{I,imp} : R_{I,imp}^{(\alpha,\beta)}(\mathbf{r}) = & \left(\mathcal{L}_v^{(\alpha)}(\hat{p}^{(\alpha)}(\mathbf{r})) - \frac{\hat{p}^{(\alpha)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} \right) \\ & + \left(\mathcal{L}_v^{(\beta)}(\hat{p}^{(\beta)}(\mathbf{r})) - \frac{\hat{p}^{(\beta)}(\mathbf{r})}{\bar{Z}_I(\mathbf{r})} \right) \end{aligned} \tag{43c}$$

7.2 Weighted residual formulation

Similar to the construction of system matrices in previous sections, for each subdomain, the pressure/normal continuity interface residuals (43b-43a) (the impedance continuity interface residual (43c) is not considered for now) are orthogonalised with respect to a weighting function $\tilde{p}$ or its derivative as

$$\begin{aligned} & \sum_{\beta=1,\beta\neq\alpha}^{N_\Omega} \int_{\Gamma_{I,v}^{(\alpha,\beta)}} \tilde{p}^{(\alpha)}(\mathbf{r}) R_{I,v}^{(\alpha,\beta)}(\mathbf{r}) \mathrm{d}\Gamma \\ & + \sum_{\beta=1,\beta\neq\alpha}^{N_\Omega} \int_{\Gamma_{I,p}^{(\alpha,\beta)}} -\mathcal{L}_v^{(\alpha)}(\tilde{p}^{(\alpha)}(\mathbf{r})) R_{I,p}^{(\alpha,\beta)}(\mathbf{r}) \mathrm{d}\Gamma = 0 \end{aligned} \tag{44}$$

The weighting function $\tilde{p}^{(\alpha)}(\mathbf{r})$ is expanded in terms of the same set of acoustic wave functions used in the pressure expansion (29)

$$\tilde{p}^{(\alpha)}(\mathbf{r}) = \sum_{a=1}^{n_w^{(\alpha)}} \tilde{p}_a^{(\alpha)} \phi_a^{(\alpha)}(\mathbf{r}) = \mathbf{\Phi}^{(\alpha)}(\mathbf{r}) \tilde{\mathbf{p}}_w^{(\alpha)} \tag{45}$$

For a multi-domain WB formulation, the above integrals in (44) are included in the weighted residual formulation (29).

7.3 System of linear equations

Substitution of the pressure expansion from Eq. (29) and the weighting function (45) for subdomain $\Omega^{(\alpha)}$ into the weighted residual formulation (44) and (35) from previous section yields a slightly more complicated system of equations compared to (37), since we now have to include formulations for the N_Ω number of coupled WB subdomains:

$$\begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{C}^{(1,2)} & \mathbf{C}^{(1,3)} & \dots & \mathbf{C}^{(1,N_\Omega)} \\ \mathbf{C}^{(2,1)} & \mathbf{A}^{(2)} & \mathbf{C}^{(2,3)} & \dots & \mathbf{C}^{(2,N_\Omega)} \\ & & \vdots & & \\ \mathbf{C}^{(N_\Omega,1)} & \mathbf{C}^{(N_\Omega,2)} & \mathbf{C}^{(N_\Omega,3)} & \dots & \mathbf{A}^{(N_\Omega)} \end{bmatrix} \begin{bmatrix} \mathbf{p}_w^{(1)} \\ \mathbf{p}_w^{(2)} \\ \vdots \\ \mathbf{p}_w^{(N_\Omega)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(N_\Omega)} \end{bmatrix} \tag{46}$$

where

$\mathbf{A}^{(\alpha)} = \mathbf{A}_p^{(\alpha)} + \mathbf{A}_v^{(\alpha)} + \mathbf{A}_Z^{(\alpha)} + \mathbf{A}_{Ip}^{(\alpha)} + \mathbf{A}_{Iv}^{(\alpha)}$ $(n_w^{(\alpha)} \times n_w^{(\alpha)})$ with

$$\mathbf{A}_{Ip}^{(\alpha)} = -\int_{\Gamma_{Ip}^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \mathbf{\Phi}^{(\alpha)} \mathrm{d}\Gamma \tag{47}$$

$$\mathbf{A}_{Iv}^{(\alpha)} = \int_{\Gamma_{Iv}^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{\Phi}^{(\alpha)T} \mathbf{n}^{(\alpha)T} \mathbf{B}^{(\alpha)} \mathrm{d}\Gamma \tag{48}$$

$$\tag{49}$$

$\mathbf{C}^{(\alpha,\beta)}$ $(n^\alpha \times n^\beta)$, is a non-zero matrix only if the two domains $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ are adjacent and share an interface

$$\begin{aligned} \mathbf{C}^{(\alpha,\beta)} = &\int_{\Gamma_{I,p}^{(\alpha,\beta)}} \frac{j}{p\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \mathbf{\Phi}^{(\beta)} \mathrm{d}\Gamma \\ &+ \int_{\Gamma_{I,v}^{(\alpha,\beta)}} \frac{j}{p\omega} \mathbf{\Phi}^{(\alpha)T} \mathbf{n}^{(\beta)T} \mathbf{B}^{(\beta)} \mathrm{d}\Gamma \end{aligned} \tag{50}$$

$\mathbf{f}^{(\alpha)} = \mathbf{f}_v^{(\alpha)} + \mathbf{f}_p^{(\alpha)} + \mathbf{f}_Z^{(\alpha)} + \mathbf{f}_{Ip}^{(\alpha)} - \mathbf{f}_{Iv}^{(\alpha)}$ $(n_w^{(\alpha)} \times 1)$

$$\mathbf{f}_{Iv}^{(\alpha)} = \int_{\Gamma_{Iv}^{(\alpha)}} \mathbf{\Phi}^{(\alpha)T} \left(\frac{j}{\rho\omega} \mathbf{n}^{(\alpha)T} \nabla \hat{p}_q^{(\alpha)} \right) \mathrm{d}\Gamma \tag{51}$$

$$\mathbf{f}_{Ip}^{(\alpha)} = \int_{\Gamma_{Ip}^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \hat{p}_q^{(\alpha)} \mathrm{d}\Gamma \tag{52}$$

$\mathbf{f}^{(\alpha,\beta)}$ $(n_w^{(\alpha)} \times 1)$, is a non-zero matrix only if the two domains $\Omega^{(\alpha)}$ and $\Omega^{(\beta)}$ are adjacent and share an interface

$$\begin{aligned} \mathbf{f}^{(\alpha,\beta)} = &-\int_{\Gamma_{Ip}^{(\alpha)}} \frac{j}{\rho\omega} \mathbf{B}^{(\alpha)T} \mathbf{n}^{(\alpha)} \hat{p}_q^{(\beta)} \mathrm{d}\Gamma \\ &- \int_{\Gamma_{Iv}^{(\alpha)}} \mathbf{\Phi}^{(\alpha)T} \left(\frac{j}{\rho\omega} \mathbf{n}^{(\beta)T} \nabla \hat{p}_q^{(\beta)} \right) \mathrm{d}\Gamma \end{aligned} \tag{53}$$

$\mathbf{b}^{(\alpha)}$ dimension $(n_w^{(\alpha)} \times 1)$ being the force

$$\mathbf{b}^{(\alpha)} = \mathbf{f}^{(\alpha)} + \sum_{\beta=1,\beta\neq\alpha}^{N_\Omega} \mathbf{f}^{(\alpha,\beta)} \tag{54}$$

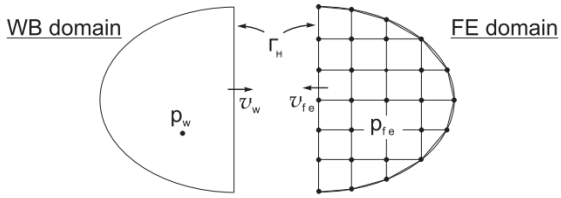


Figure 6: FE and WB domain coupling. Figure from [**?**].

We write it in compact form as

$$\mathbf{A}\mathbf{p}_w = \mathbf{b} \tag{55}$$

For more details see [**?**] p. 67-70.

8 The Hybrid FE-WBM

For a direct pressure and velocity coupling at the interface Γ_H between the FEM and the WBM are given similar to Eq. (41b-41a) (see Figure 6

$$r \in \Gamma_H : v_{fe}(\mathbf{r}) = -v_w(\mathbf{r}) \tag{56}$$
$$r \in \Gamma_H : p_w(\mathbf{r}) = p_{fe}(\mathbf{r}) \tag{57}$$

8.1 Formulation of residuals

The interface residuals error functions are given as

$$r \in \Gamma_{I,v} : R_{H,v}^{(w,fe)}(\mathbf{r}) = \mathcal{L}_v^{(w)}(\hat{p}_w(\mathbf{r})) + \mathcal{L}_v^{(fe)}(\hat{p}_{fe}(\mathbf{r})) \tag{58}$$
$$r \in \Gamma_{I,p} : R_{H,p}^{(w,fe)}(\mathbf{r}) = \hat{p}_w(\mathbf{r}) - \hat{p}_{fe}(\mathbf{r}) \tag{59}$$

remembering that the normal velocity operator $\mathcal{L}_v$ is defined as

$$\mathcal{L}_v^{(fe)}(\bullet) = \frac{j}{\rho\omega}\frac{\partial \bullet}{\partial n_{fe}} = \frac{j}{\rho\omega}(\mathbf{n}_{fe})^T\nabla\bullet$$
$$\mathcal{L}_v^{(w)}(\bullet) = \frac{j}{\rho\omega}\frac{\partial \bullet}{\partial n_{w}} = \frac{j}{\rho\omega}(\mathbf{n}_{w})^T\nabla\bullet$$

with $\mathbf{n}_{fe} = [n_{fe_x}, n_{fe_y}]$ and $\mathbf{n}_w = [n_{w_x}, n_{w_y}]$ being the normal to the FE and WE interface, respectively.

8.2 Weighted residual formulation

The pressure continuity condition Eq. (56) is imposed on the WB boundary as the pressure boundary condition and is added to the weighted residual formulation Eq. (35)

$$\int_{\Gamma_{H,p}} -\mathcal{L}_v^{(w)}(\tilde{p}_w(\mathbf{r}))R_{H,p}^{(w,fe)}(\mathbf{r})\mathrm{d}\Gamma \tag{60}$$

and the velocity continuity condition Eq. (57) is imposed on the boundary of the FE domain as the velocity boundary condition and is added to the weighted residual formulation Eq. (35)

$$\int_{\Gamma_{H,v}} -\tilde{p}_{fe}(\mathbf{r})R_{H,v}^{(w,fe)}(\mathbf{r})\mathrm{d}\Gamma \tag{61}$$

where each of the weighting functions $\tilde{p}_{wb}(\mathbf{r})$ and $\tilde{p}_{fe}(\mathbf{r})$ are expanded with the same set of acoustic wave functions used in the pressure expansion as

$$\tilde{p}_{fe}^{(\alpha)}(\mathbf{r}) = \sum_{a=1}^{n_{fe}^{(\alpha)}} \tilde{p}_{fe,a}^{(\alpha)} N_a^{(\alpha)}(\mathbf{r}) = \mathcal{N}^{(\alpha)}(\mathbf{r}) \tilde{\mathbf{p}}_{fe}^{(\alpha)} \tag{62a}$$

$$\tilde{p}_{wb}^{(\alpha)}(\mathbf{r}) = \sum_{a=1}^{n_{w}^{(\alpha)}} \tilde{p}_{a}^{(\alpha)} \phi_a^{(\alpha)}(\mathbf{r}) = \mathbf{\Phi}^{(\alpha)}(\mathbf{r}) \tilde{\mathbf{p}}_{w}^{(\alpha)} \tag{62b}$$

For more details see [**?**] p. 204-205).

8.3 System of linear equations

The following equation system is constructed

$$\begin{bmatrix} \mathbf{S_{fe}} & \mathbf{Q_{fw}} \\ \mathbf{Q_{wf}} & \mathbf{S_w} \end{bmatrix} \begin{Bmatrix} \mathbf{p_{fe}} \\ \mathbf{p_w} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f_{fe}} + \mathbf{f_{fw}} \\ \mathbf{b_w} + \mathbf{c_b} \end{Bmatrix} \tag{63}$$

with $n_{tot} = n_{fe} + n_w$ degrees of freedom and

$$\begin{aligned}
\mathbf{Q}_{wf} &= \int_{\Gamma_H} \frac{j}{\rho\omega} \mathbf{B}^T \mathbf{n}_w \mathcal{N}_{fe} \mathrm{d}\Gamma, && (n_w \times n_{fe}) \text{ coupling matrix} \\
\mathbf{Q}_{fw} &= \int_{\Gamma_H} (\mathcal{N}_{fe})^T \cdot (\mathbf{n}_w)^T \mathbf{B} \mathrm{d}\Gamma && (n_{fs} \times n_w) \text{ coupling matrix} \\
\mathbf{C}_b &= -\int_{\Gamma_H} \frac{j}{\rho\omega} \mathbf{B}^T \mathbf{n}_w \mathbf{\Phi} \mathrm{d}\Gamma, && (n_w \times n_{fe}) \text{ WB back-coupling matrix} \\
\mathbf{c}_b &= \int_{\Gamma_H} \frac{j}{\rho\omega} \mathbf{B}^T \mathbf{n}_w \mathrm{d}\Gamma, && (n_w \times 1) \text{ WB back-coupling vector} \\
\mathbf{f}_{fw} &= -\int_{\Gamma_H} (\mathcal{N}_{fe})^T \cdot (\mathbf{n}_w)^T \nabla \hat{p}_q \mathrm{d}\Gamma && (n_{fe} \times 1) \text{ loading vector on the FE} \\
\mathbf{S_w} &= \mathbf{A} + \mathbf{C_b} && (n_w \times n_w) \text{ system and back-coupling matrix}
\end{aligned}$$

For more details, see [**?**] p. 205 and [**?**].

9 Solution of the coupled model

The system of equation in Eq. (63) is populated as follows

- $\mathbf{S}_{fe}$ is a typical symmetric, sparsely populated banded structured FE matrix
- $\mathbf{S}_w$ results from the WB system matrix and the WB back-coupling matrices $\mathbf{S}_w = \mathbf{A} + \mathbf{C_b}$. Densely populated, but in general much smaller than $\mathbf{S}_{fe}$.
- $\mathbf{Q}_\bullet$ are sparsely populated rectangular matrices since the matrices are non-zero only if two WB-FE subdomains are adjacent.

A three-step procedure has been proposed:

1. Isolate $\mathbf{p}_{fe}$ from the top part of matrix equation (63)

$$\mathbf{p}_{fe} = \mathbf{S}_{fe}^{-1}(\mathbf{s}_{fe} - \mathbf{Q}_{fw}\mathbf{p}_w) \tag{64}$$

2. Substitute $\mathbf{p}_{fe}$ into the bottom part of the matrix equation (63)

$$(-\mathbf{Q}_{wf}\mathbf{S}_{fe}^{-1}\mathbf{Q}_{fw} + \mathbf{S}_w)p_w = \mathbf{s}_w - \mathbf{Q}_{wf}\mathbf{S}_{fe}^{-1}\mathbf{s}_{fe} \tag{65}$$

3. Introduce two sparse linear systems which can be solved efficiently for H and h using sparse solvers

$$\mathbf{S}_{fe}\mathbf{H} = \mathbf{Q}_{fw} \tag{66a}$$
$$\mathbf{S}_{fe}h = \mathbf{s}_{fe} \tag{66b}$$

   Note that the above equations have identical left-hand side matrices, allowing LU decomposition solving both equations at the same time.

4. Rewrite Eq. (65) in terms of H and h and solve for p_w with a dense solver

$$(-\mathbf{Q}_{wf}\mathbf{H} + \mathbf{S}_w)p_w = \mathbf{s}_w - \mathbf{Q}_{wf}\mathbf{h} \tag{67}$$

5. The eliminated p_{fe} nodal dofs are retrieved by matrix multiplications. Isolating H and h in Eq. (66a) and (66b), respectively, and inserting into (64) yields

$$\mathbf{p}_{fe} = -\mathbf{H}\mathbf{p}_w + \mathbf{h}$$